

University of Warwick institutional repository: <http://go.warwick.ac.uk/wrap>

A Thesis Submitted for the Degree of PhD at the University of Warwick

<http://go.warwick.ac.uk/wrap/45832>

This thesis is made available online and is protected by original copyright.

Please scroll down to view the document itself.

Please refer to the repository record for this item for information to help you to cite it. Our policy information is available from the repository home page.

Library Declaration and Deposit Agreement

1. STUDENT DETAILS

Please complete the following:

Full name:

University ID number:

2. THESIS DEPOSIT

2.1 I understand that under my registration at the University, I am required to deposit my thesis with the University in BOTH hard copy and in digital format. The digital version should normally be saved as a single pdf file.

2.2 The hard copy will be housed in the University Library. The digital version will be deposited in the University's Institutional Repository (WRAP). Unless otherwise indicated (see 2.3 below) this will be made openly accessible on the Internet and will be supplied to the British Library to be made available online via its Electronic Theses Online Service (EThOS) service.

[At present, theses submitted for a Master's degree by Research (MA, MSc, LLM, MS or MMedSci) are not being deposited in WRAP and not being made available via EThOS. This may change in future.]

2.3 In exceptional circumstances, the Chair of the Board of Graduate Studies may grant permission for an embargo to be placed on public access to the hard copy thesis for a limited period. It is also possible to apply separately for an embargo on the digital version. (Further information is available in the *Guide to Examinations for Higher Degrees by Research*.)

2.4 *If you are depositing a thesis for a Master's degree by Research, please complete section (a) below. For all other research degrees, please complete both sections (a) and (b) below:*

(a) Hard Copy

I hereby deposit a hard copy of my thesis in the University Library to be made publicly available to readers (please delete as appropriate) EITHER immediately OR after an embargo period of months/years as agreed by the Chair of the Board of Graduate Studies.

I agree that my thesis may be photocopied.

YES / NO (*Please delete as appropriate*)

(b) Digital Copy

I hereby deposit a digital copy of my thesis to be held in WRAP and made available via EThOS.

Please choose one of the following options:

EITHER My thesis can be made publicly available online. YES / NO (*Please delete as appropriate*)

OR My thesis can be made publicly available only after.....[date] (*Please give date*)

YES / NO (*Please delete as appropriate*)

OR My full thesis cannot be made publicly available online but I am submitting a separately identified additional, abridged version that can be made available online.

YES / NO (*Please delete as appropriate*)

OR My thesis cannot be made publicly available online.

YES / NO (*Please delete as appropriate*)

3. GRANTING OF NON-EXCLUSIVE RIGHTS

Whether I deposit my Work personally or through an assistant or other agent, I agree to the following:

Rights granted to the University of Warwick and the British Library and the user of the thesis through this agreement are non-exclusive. I retain all rights in the thesis in its present version or future versions. I agree that the institutional repository administrators and the British Library or their agents may, without changing content, digitise and migrate the thesis to any medium or format for the purpose of future preservation and accessibility.

4. DECLARATIONS

(a) I DECLARE THAT:

- I am the author and owner of the copyright in the thesis and/or I have the authority of the authors and owners of the copyright in the thesis to make this agreement. Reproduction of any part of this thesis for teaching or in academic or other forms of publication is subject to the normal limitations on the use of copyrighted materials and to the proper and full acknowledgement of its source.
- The digital version of the thesis I am supplying is the same version as the final, hard-bound copy submitted in completion of my degree, once any minor corrections have been completed.
- I have exercised reasonable care to ensure that the thesis is original, and does not to the best of my knowledge break any UK law or other Intellectual Property Right, or contain any confidential material.
- I understand that, through the medium of the Internet, files will be available to automated agents, and may be searched and copied by, for example, text mining and plagiarism detection software.

(b) IF I HAVE AGREED (in Section 2 above) TO MAKE MY THESIS PUBLICLY AVAILABLE DIGITALLY, I ALSO DECLARE THAT:

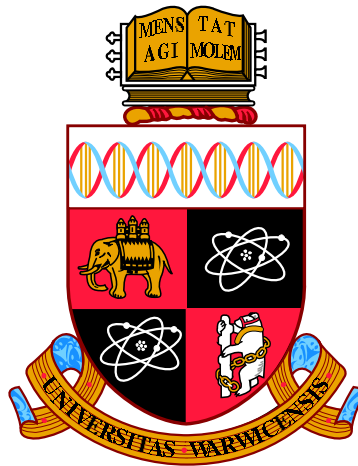
- I grant the University of Warwick and the British Library a licence to make available on the Internet the thesis in digitised format through the Institutional Repository and through the British Library via the EThOS service.
- If my thesis does include any substantial subsidiary material owned by third-party copyright holders, I have sought and obtained permission to include it in any version of my thesis available in digital format and that this permission encompasses the rights that I have granted to the University of Warwick and to the British Library.

5. LEGAL INFRINGEMENTS

I understand that neither the University of Warwick nor the British Library have any obligation to take legal action on behalf of myself, or other rights holders, in the event of infringement of intellectual property rights, breach of contract or of any other right, in the thesis.

Please sign this agreement and return it to the Graduate School Office when you submit your thesis.

Student's signature: Date:



Games for Optimal Controller Synthesis on Hybrid Automata

by

Michał Rutkowski

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Department of Computer Science

September 2011

THE UNIVERSITY OF
WARWICK

Contents

List of Figures	iv
List of Tables	vii
Acknowledgments	viii
Declarations	ix
Abstract	x
Chapter 1 Introduction	1
1.1 Motivation	1
1.2 Literature review	5
1.2.1 Models	8
1.2.2 Qualitative analysis	10
1.2.3 Quantitative analysis	16
1.3 Contributions	23
1.4 Organisation	28
Chapter 2 Preliminaries	30
2.1 Definability and decidability	31
2.2 Zero-sum games	33
2.2.1 Games in strategic form	34

2.2.2	Games on graphs	37
2.3	Hybrid automata	62
2.3.1	Hybrid automata with strong resets	69
2.3.2	Timed automata	71
2.3.3	Games on hybrid automata	78
Chapter 3	Games on single-clock timed automata	83
3.1	Timed automata and reachability-price games	84
3.1.1	Simplifications	89
3.1.2	Operations	92
3.2	Cost functions	95
3.3	Solving reachability-price games on simple timed automata	102
3.3.1	Computing the value of reachability-price games on single- clock timed automata	104
3.3.2	Correctness and complexity	122
Chapter 4	Games on price-reward graphs	140
4.1	Price-reward game graphs	141
4.1.1	Average-price-per-reward games	146
4.1.2	Optimality equations	148
4.2	Finite average-price-per-reward games	155
4.2.1	Finite game graphs of out-degree one	155
4.2.2	The general case	158
Chapter 5	Games on hybrid automata with strong resets	162
5.1	Games on hybrid automata with strong resets	163
5.1.1	Hybrid average-price-per-reward games	172
5.1.2	Hybrid reachability-price games	178
5.2	Solving hybrid games with strong resets	184

5.2.1	A finite abstraction	184
5.2.2	Solving hybrid average-price-per-reward games	191
5.2.3	Solving hybrid reachability-price games	198
Chapter 6	Conclusions	204
Appendix A	Proof of Theorem 3.10	207

List of Figures

1.1	Basic rules of the boiler's operation. The invariant is that the temperature of the water never drops below zero, and never exceeds 86.4°C.	2
1.2	Heating and cooling specification of the boiler.	3
2.1	A simplified game graph that models a production process in a factory. Each edge has a price (written above the edge) and a reward (written below the edge); we have omitted those that are equal to zero. Circular vertices belong to player Min; the remaining vertices belong to player Max.	51
2.2	Production model from Figure 2.1. Each state is annotated with the value of its bias, under the assumption that the gain of all vertices is $\frac{5}{6}$. Bias of a state is computed based on the bias of the successor state. When choice of successors was available, the solid line highlights the choice made.	53
2.3	A simple game graph that models a single iteration of a production process. The production process starts in the "Start" state, and if all goes well, finishes in the "Sale" state. Edges are annotated with their prices, unless it is equal to zero; the rewards have been omitted. Circle states belong to player Min, and the remaining states belong to player Max. The "Sale" state is the goal state only.	57

2.4	The game graph of Figure 2.3. The states are partitioned into the sets W^{Max} , W^{Min} , and $S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$. The states in the $S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$ are annotated with the values of the P and D functions, which satisfy the reachability-price optimality equations.	61
2.5	Hybrid automaton that models the boiler.	65
2.6	A graphical representation of the hybrid automaton with strong resets that models the boiler, as introduced in Example 2.37.	72
2.7	Timed automaton that models the boiler.	78
2.8	Hybrid automaton with multiple trajectories (diagram a)) and with a single trajectory (diagram b)).	82
3.1	a) Cost function f , before applying $\min C(f, c)$ operator — dashed lines have a slope $-c$; b) cost function $g = \min C(f, c)$ — dashed lines denote parts of f that do not coincide with g	98
3.2	a) Cost function f , before applying $\max C(f, c)$ operator — dashed lines have a slope $-c$; b) cost function $h = \max C(f, c)$ — dashed lines denote parts of f that do not coincide with h	98
3.3	Iterative computation of $\text{Val}_\Gamma(\ell^*)$ over the interval $I = [b, e]$, where $\ell^* \in \mathbf{L}^{\text{Max}}$. Diagrams a) and b) depict two subsequent iterations; the grey rectangle indicates the subinterval for which the $\text{Val}_\Gamma(\ell^*)$ function is being computed during the given iteration.	113
3.4	Iterative computation of $\text{Val}_\Gamma(\ell^*)$ over the interval $I = [b, e]$, where $\ell^* \in \mathbf{L}^{\text{Min}}$. Diagrams a) and b) depict two subsequent iterations; the grey rectangle indicates the subinterval for which the $\text{Val}_\Gamma(\ell^*)$ function is being computed during the given iteration.	113

4.1	A simple price-reward game graph modelling arbitrage between two markets. All states belong to player Min. The matrices annotating the edges show how the price (reward) of an edge changes depending on players' inputs. Player Min can choose between M1 and M2, whereas player Max can choose between L and H.	144
4.2	A simple price-reward graph; all edges, except the horizontal one, have fixed prices and rewards. The price of an edge is written above it, whereas the reward is written below. The horizontal edge admits two inputs from player Min, and the set of inputs for player Max is empty. All states belong to player Min. Figure a) shows the values of the price-per-reward game from every state, depending on the chosen inputs, when player Min chooses to play "up" from the grey state. Likewise, Figure b) shows the respective values when player Min chooses to play "down" from the grey state.	152
5.1	The hybrid automaton \mathcal{H} discussed in Example 5.11. a) Graph structure underlying \mathcal{H} . b) State space of \mathcal{H} . c) Guard function of \mathcal{H} . . .	178
5.2	Numbers 1,2, and 3 represent the three steps of a game round in a hybrid game with strong resets. a) the order in which the steps are carried out in the game Γ . b) the order in which the steps are carried out in the game $\hat{\Gamma}$	187
5.3	The price-reward game graph $\hat{\Gamma}$ obtained, from Γ introduced in Example 5.11, using the finite abstraction \sim . A_1 stands for $\{a\}$ and A_2 for $\{a, b\}$. Edge price is omitted when it is equal to zero. Edges are annotated with constant prices (if omitted, it is equal to 0); the rewards are omitted, as they are constant and equal to 1.	192
5.4	Solid arrows denote the optimal strategies of both players. Above each vertex one can find its gain and bias.	199

List of Tables

2.1	Price-per-reward ratios for a single production cycle of Figure 2.1.	50
2.2	Costs of reaching the “Sale” state from the “Start” state in the game graph of Figure 2.3.	56

Acknowledgments

First and foremost I would like to thank my supervisors Dr. Marcin Jurdziński and Dr. Ranko Lazić who made it possible for me to undertake my Ph.D. studies at the University of Warwick. I would like to thank them for all the inspiration and valuable feedback they gave me, and for being there at times of crisis.

I would like to thank my parents Jan and Zofia, my sister Marysia, and my wife Agnieszka for being a wonderful and supportive family with whom I could share my worries and joys.

My two friends John Fearnley and Ashutosh Trivedi deserve special thanks. They have provided me with valuable feedback, and our discussions have been very helpful and inspiring.

I would also want to thank all of my colleagues from Warwick University: Rosario Undurraga Riesco, Daniel Klaus, Noelia Alvares Garcia, Elisabeth Simbuerger, Ritesh Krishna, YinYin Yuan, Daniel Valdez Amaro, Soledad Betanzos Lara, Cathleen Colard, Cherie Wang, Jared Sulem, Evelyn Sulem, Rodrigo Cordero Vega, Soledad Pinto Sanchez, Michał Komorowski, Haris Aziz, Sana Aziz, Helena Stec and Massimo Perufo. Without you, my time at Warwick University would not have been as enjoyable!

Finally, I would like to thank my dearest Polish friends: Tuśka, Bocian, Justa, Paweł, Ania Naj, Mateo, Borys, Ola, Marcin Karol, Natalia, Klaudyna, and Magda. Although we have been far apart, we managed to keep in touch. Our Skype conversations, joint trips, and even one Easter Holiday, were the source of much needed positive energy.

Declarations

The work presented in Chapter 4 is a joint work with my supervisors Dr. Marcin Jurdziński and Dr. Ranko Lazić. The work presented in Chapter 5 is a joint work with Dr. Patricia Bouyer-Decitre, Dr. Thomas Brihaye, and my supervisors Dr. Marcin Jurdziński and Dr. Ranko Lazić. My co-authors contributed in the high-level ideas behind the models and some of the techniques used. In both cases, the results and proofs presented in this thesis are my own. The work presented in Chapter 3 is my own, however, for completeness sake we provide some results that were already present in the literature — such cases are clearly marked.

This thesis is written by myself, under the supervision of Dr. Marcin Jurdziński and Dr. Ranko Lazić. None of the work presented in this thesis was submitted for a previous degree, or a degree at a different university. Preliminary versions of the work presented in this thesis have been published, or have been accepted for publication: the work presented in Chapter 4 has been published in VMCAI 2010 [JLR09], and in the STTT Journal [RLJ10]; the work presented in Chapter 5 has been published in FORMATS 2008 [BBJ⁺08], VMCAI 2010 [JLR09], and in the STTT Journal [RLJ10]; the work presented in Chapter 3 has been published in QAPL 2011 [Rut11].

Abstract

Hybrid automata are an extension of finite automata obtained by augmenting them with a set of real-valued variables. Hybrid automata are a natural model for hybrid systems, i.e., systems that exhibit both discrete and continuous behaviour. Such systems naturally arise when studying the interaction of a digital controller with an analogue plant. In this thesis we study the problem of optimal controller synthesis for such systems. The goal is to design a control program that will ensure that a system behaves optimally regardless of some uncontrollable external factors. We study this problem from the perspective of two-player games on hybrid automata.

In this thesis two variants of hybrid automata are being considered: single-clock timed automata and hybrid automata with strong resets. The first model allows us to capture the passage of time between discrete events, and the other model allows us to capture the more complex dynamics the hybrid systems.

The controller synthesis problem is modelled as follows. There are two players: the controller and the adversarial environment. Their interaction is viewed as a multi-stage process, in which the two players interact to create an execution of the hybrid automaton. The problem of synthesising an optimal controller is: given a certain optimisation objective, compute an “optimal” strategy for the controller player. Executions of the hybrid automaton are assigned two quantities: the price and the reward. Two optimisation objectives are considered: price-per-reward ratio of an infinite execution and the price of reaching a certain state of the hybrid automaton (reachability-price).

In this thesis we prove that optimal controllers for the reachability-price objective can be synthesised on single-clock timed automata in EXPTIME. Furthermore, we prove that optimal controllers for reachability-price and average-price-per-reward objectives can be synthesised on hybrid automata with strong resets.

Chapter 1

Introduction

In this thesis we study games for optimal controller synthesis on hybrid automata. In particular we study average-price-per-reward and reachability-price games on hybrid automata with strong resets, and reachability-price games on single-clock timed automata. We start this chapter with a gentle introduction to hybrid automata and games for controller synthesis. Next, we provide a brief survey of the related literature, and we then discuss the contributions of this thesis. We conclude this chapter with an outline of the remainder of the thesis.

1.1 Motivation

Hybrid systems are systems that exhibit both continuous and discrete behaviour. Such systems naturally arise when modelling an interaction of a digital controller with an analogue environment [NOSY93; ACHH93; Hen96; HKPP98]. There are various examples of hybrid systems, e.g., audio protocols [HWT95], industrial steam boilers [HWT96], semi-automated vehicles [ADG97], aircraft landing systems [TMBO03], and user interfaces [OMBT03].

A very basic example of a hybrid system is a thermostat controlling a boiler. The discrete component of the system is its mode of operation, which is either

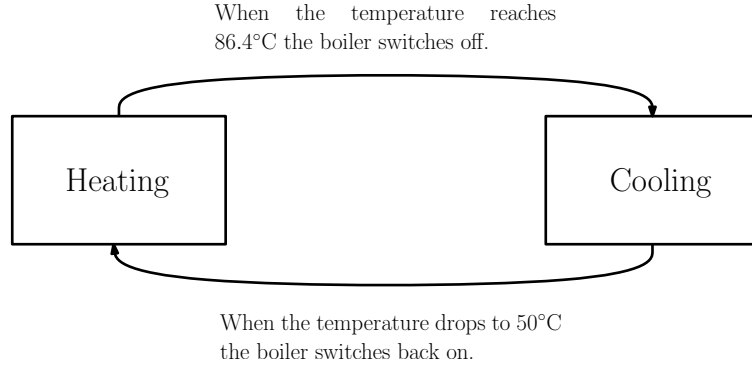


Figure 1.1: Basic rules of the boiler's operation. The invariant is that the temperature of the water never drops below zero, and never exceeds 86.4°C .

heating or cooling, and the continuous component of the system is its temperature. The mode of operation together with the current temperature form the state of this hybrid system. A change of mode is guided by the current temperature of the water stored. The mode of operation together with the current temperature determine the state of this hybrid system. Figure 1.1 provides a schematic description of the boiler. When in heating mode, once the water reaches the temperature of 86.4°C the boiler switches off, and enters the cooling mode. When in the cooling mode, on the other hand, once the temperature of the water drops to 50°C the boiler goes back to the heating mode. The heater in the boiler can heat the water to a maximum 86.4°C , and the water never freezes. The heating and cooling characteristics of the boiler, i.e., the laws according to which the temperature changes can be seen in Figure 1.2. If the boiler is in the heating mode, and the initial temperature of the water is 0°C then it takes 120 minutes for the water to reach its maximum temperature of 86.4°C .

In this thesis we study the problem of optimal controller synthesis for hybrid systems. The goal is to design a control program that will ensure that the system behaves optimally regardless of some uncontrollable external factors. For example, consider a battery-powered portable device with a hard drive. Operations which require accessing the hard drive consume a lot of energy, which is a valuable resource. In this case optimisation would mean ensuring that, regardless of the users actions,

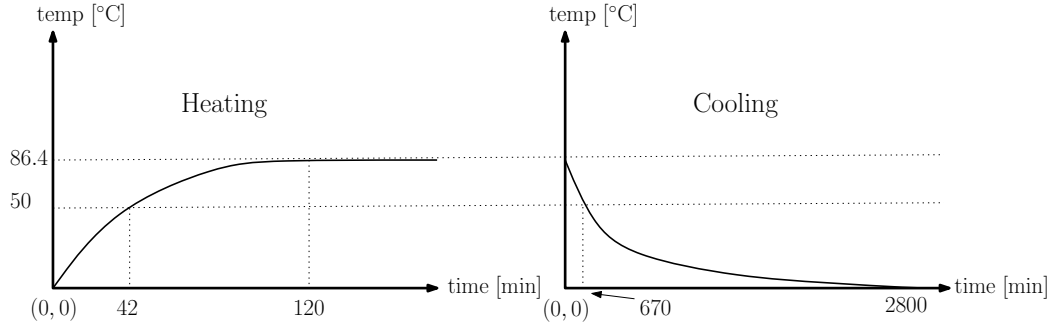


Figure 1.2: Heating and cooling specification of the boiler.

the disk operations are executed by the controller in such a way that the energy consumption is minimised. In this thesis we will be considering worst-case optimisation. That is, we will want to ensure optimal performance when the external factor plays to our disadvantage.

In our approach, a hybrid system is modelled using a hybrid automaton. Hybrid automata are an extension of finite automata, obtained by augmenting them with a set of real-valued variables. The finite automaton models the discrete component, and the real-valued variables model the continuous component. In this thesis, we will be considering two variants of hybrid automata: single-clock timed automata and hybrid automata with strong resets. The first model allows to capture the passage of time between discrete events, and the other model allows to capture more complex dynamics present in hybrid systems. However, in the latter model, the continuous component of the automaton's state has to be reset after each discrete event.

We model the controller synthesis problem as a two player game. There are two players, the controller and the adversarial environment. The worst-case scenario approach is captured by the fact that we are considering zero-sum games, i.e., games in which the winnings of one player are equal to the losses of the other. We view players' interaction as a multi-stage process, in which the two players create an execution of the hybrid automaton. Every execution of the automaton

will be assigned two quantities, its price and its reward. Given a certain optimisation objective, to synthesise an optimal controller will mean to compute an “optimal” strategy for the controller. A strategy can be viewed as a rule that tells the player how to interact with the other player at every stage of the process.

We will be considering two optimisation objectives. Firstly, we will focus our attention to the average-price-per-reward objective, i.e., the limit ratio of accumulated price and accumulated reward of the execution will be optimised. In this case we are dealing with recurring behaviour and optimising average performance. Consider the following example. There is a production plant which, once started, can operate for long periods of time. The plant operates in production cycles. In every production cycle, fuel for the plant has to be bought so that the plant can manufacture. The manufactured products are sold by the company owning the plant and bring revenue. However, the revenue is diminished by the cost of the fuel needed by the plant. Fuel consumption and the sale price of the manufactured product may vary over time, but we would like to ensure that on average, the cost of fuel used by the plant is relatively small compared to the sale price of the product, i.e., that the profit of the company is as high as possible. It is natural to model this situation as a game with average-price-per-reward objective. The controller wants to minimise the limit ratio of the total cost of fuel used over the total revenue obtained. The fact that the plant operates over long periods of time is reflected by the fact that we are considering infinite executions.

Secondly, we will direct our attention to the reachability-price objective, i.e., the price of reaching a certain state will be optimised. In this case we are optimising a single execution of the system. Consider the following example. A plane is flying from one city to another. The main cost of such a journey is the cost of the fuel used. There are various weather factors that affect the duration of the journey, and hence the amount of fuel consumed. The exposure to the weather factors depends on the route taken; sometimes avoiding this exposure may result in taking an overly

long route. We want to design an autopilot (controller) which chooses a flight plan that minimises the fuel consumption. Such a situation is naturally modelled by a game with a reachability-price objective. In this case the price of an execution is the amount of fuel used. Notice that in this case we are not interested in the reward of an execution.

In the examples above, the optimisation objective was to minimise a certain quantity. However, the framework of zero-sum games is equally suitable for optimisation objectives where one wants to maximise that quantity. In particular, we could consider average-price-per-reward and reachability-price objectives where the goal would be to maximise those quantities.

In this thesis we prove that optimal controllers for reachability-price can be synthesised on single-clock timed and hybrid automata with strong resets. Furthermore, we prove that optimal controllers for average-price-per-reward can be synthesised on hybrid automata with strong resets.

1.2 Literature review

In this section we survey the literature related to the subject of games for optimal controller synthesis for hybrid systems. This review is by no means exhaustive. We concentrate on results that are of relevance to the problems considered in this thesis. The purpose of this review is to get the reader familiar with different subclasses of hybrid systems. In particular, we want the reader to understand how timed automata and o-minimal hybrid automata fit in the general theory of hybrid automata. This will enable the reader to appreciate how our results contribute to that theory.

We start this section with a summary of our approach to controller synthesis, then we proceed to present the various subclasses of hybrid automata, and conclude with a survey of results. This survey should provide the understanding of

the differences between the different subclasses of hybrid automata.

Hybrid automata. Hybrid systems are systems that exhibit both continuous and discrete behaviours. Alur et al. [ACHH93] and Nicolin et al. [NOSY93], independently, proposed *hybrid automata* as a model for such systems. A hybrid automaton is an extension of a finite automaton obtained by augmenting it with a set of real-valued variables. In this context, the finite automaton is referred to as a *control graph*. A vertex of the automaton, referred to as the *control location*, models the state of the digital controller and the valuation of the variables models the state of the analogue environment. Together, this pair forms the state of the hybrid automaton. A *flow predicate* describes the admissible changes (flows, trajectories) to the value of the variables within a given control location. Changes of control locations are governed by edges of the control graph, referred to as *control actions*. Availability of a given control action depends on the valuation of the real-valued variables, and is specified by the *jump predicate*.

Alur et al. and Nicolin et al. proposed transition semantics for this model. A state of the transition system is a pair consisting of the current control location and the current valuation of the real-valued variables. There are two kinds of transitions. Discrete transitions, which correspond to changes of the current control locations, are instantaneous and are labelled with the corresponding control action. Continuous transitions, which correspond to the changes to the current valuation of the real-valued variables, span over time and are labelled with their time duration. A transition results in a change to the current state of the transition system. The behaviour (execution) of a hybrid system is thus modelled by a run of the transition system.

Controller synthesis. The goal of the controller synthesis problem is to *synthesise a control program* that restricts the systems behaviour in such a way that its every execution satisfies certain properties [Chu62; RW89]. The controller synthesis

problem was first posed by Church [Chu62], and later considered by both computer scientists [Tho95; GTW02; dA03] and control theorists [RW89; FV97]. Ramadge and Wonham [RW89; Tho95] proposed modelling this problem using a two-player game on a graph.

In this setting, the game is played on a transition system by two players: Min and Max, who model the controller and the adversarial environment, respectively. The game is played in rounds and in each round both players interact to determine the next transition to be executed by the transition system [RW89; HHM99; dAFH⁺03]. A game is said to be turn-based, if the states of the transition system are partitioned between the two players, and in every state the owner of that state determines the next transition to execute [GTW02; AG11].

A play of the game is a (possibly infinite) sequence of transitions. A payoff function assigns a value to every such play — an amount of player Max’s winnings, which is equal to the player Min’s losses. These kind of games are called zero-sum games. This approach models the situation in which the controller is facing an adversarial environment, i.e., the worst case analysis [AdAF05].

A strategy of a player is a rule which guides the player in the game, i.e., tells the player what choices to make when interacting with the other player. The value of the strategy is the value of the winnings which this strategy guarantees. The lower value of the game is the value of the “best” strategy for player Max. Intuitively, the lower value is the amount of winnings which player Max can guarantee regardless, of the behaviour of player Min. The upper value is defined similarly in terms of the “best” strategy of player Min. If the lower and upper values are equal then we say that the game is determined and it has a value. Not all games are determined [dAFH⁺03].

To synthesise a controller means to compute an (almost) optimal strategy for the player who models the controller. The goal of controller synthesis is to synthesise real control programs, therefore the computed strategies have to be physically

meaningful [CHR02; dAFH⁺03; AdAF05; BHPR07]. Zeno runs are an example of runs that can be valid executions of the hybrid automaton, but have no physical meaning [dAFH⁺03]. A Zeno-run is a run consisting of infinitely many discrete transitions whose overall duration is finite, i.e., the accumulated time of all the transitions converges. There are examples of games, where one of the players, to achieve optimality, may have to enforce a Zeno run [dAFH⁺03; AdAF05; BHPR07]. De Alfaro et al. [dAFH⁺03] and Brihaye et al. [BHPR07] considered games where players are not allowed to cause Zeno runs. However, in this setting the games are not always determined. Unfortunately, even if we restrict the players to use only strategies that do not cause Zeno runs, some of these strategies may not be implementable. Cassez et al. [CHR02] gave an example of a controller which has to execute control actions after times $(0, 1/2, 1/3, 1/4, \dots, 1/n, \dots)$. Such a controller does not cause Zeno behaviour, yet its behaviour is not implementable. As a solution, the authors propose using discrete time, i.e., the controller can execute discrete actions only at integer time points.

1.2.1 Models

In this section we briefly review the different subclasses of hybrid automata that are of relevance to this thesis.

Timed automata. Timed automata were introduced by Alur and Dill [AD94]. This class of automata is characterised by two restrictions regarding the flow and jump predicates. Firstly, the real-valued variables change at the same rate as real time — that is why, in this context, the variables are referred to as *clocks*. Secondly, the predicate controlling the availability of a control action is constructed using conjunctions of the following atoms $x \leq k$ and $x - y \leq k$, where x and y are clocks, and k is a integer constant. Moreover, the only effect of a control action on the values of the clocks is that a subset of the clocks has its values reset to 0.

Linear hybrid automata Linear hybrid automata were introduced by Alur et al. [ACHH93]. Linear hybrid automata generalise timed automata by placing less restrictions on the flow and jump predicates. In this class, the jump and flow predicates are defined using linear formulae over the set of the real-valued variables. A linear formula is a boolean combination of inequalities between linear terms and integer constants, where a linear term is a linear combination of the real-valued variables and integer constants.

Rectangular hybrid automata. Rectangular hybrid automata were proposed by Alur et al. [AHH96]. Rectangular hybrid automata are more restrictive than linear hybrid automata — their flow and jump predicates are specified using rectangles, i.e., Cartesian products of intervals, rather than using arbitrary linear expressions. Rectangular hybrid automata generalise timed automata by placing less restrictions on the flow predicate, which specifies how the real-valued variables change over time; recall that in timed automata, the variables changed at a constant and uniform rate. Firstly, the rates of continuous change may vary between different variables and different control locations. Secondly, instead of a single rate per variable and location a rectangular bound, i.e., an upper and lower bound on the rate of change, is given — a one dimensional rectangle.

O-minimal hybrid automata O-minimal hybrid automata were first proposed by Lafferriere et al. [LPS00]. Their distinguishing property is that the flow and jump predicates are defined using first-order sentences over an o-minimal structure. O-minimal structures have the property that every first-order definable subset of the domain is a finite union of intervals [vdD98]. As opposed to timed, linear and rectangular hybrid automata, o-minimal hybrid automata allow for very complex flow specifications, but at the cost of strong restrictions placed on the jump predicate. The jump predicate is restricted in the following way: as a result of a discrete transition, the values of all the variables have to be non-deterministically

reinitialised to a value from a predefined set that is specific to the control action associated with the transition. We call this property *the strong reset property*.

Vladimerou et al. [VPVD08] proposed an extension of o-minimal hybrid automata, the so-called *STORMED hybrid automata*. In this new model, it is not required that the automaton has the strong reset property. However, a different set of technical restrictions is enforced: 1) the frequency of discrete transitions is bounded from above; 2) in every location and for every valuation of the real-valued variables there is only one admissible trajectory; 3) the changes to the values of the variables, which are triggered by a transition, satisfy certain monotonicity criteria; 4) the jump predicate satisfies certain boundedness criteria.

Not all the results regarding o-minimal hybrid automata are effective. Their effectiveness depends on the decidability of the first-order o-minimal theory underlying the definition of the automaton [LPS00; BM05]. Recall that given an algebraic structure, its first-order theory is the set of all sentences that are true. A theory is said to be decidable if there exists an algorithm that given a sentence in that theory decides whether it is true.

Examples of o-minimal structures include the ordered field of reals, and the ordered group of rationals [LPS00]. In particular, by Tarski's well known result [Tar51], the first-order theory of the ordered field of reals is decidable.

1.2.2 Qualitative analysis

We survey the results regarding controller synthesis for reachability on hybrid automata. The problems considered are qualitative, i.e., there is no optimisation involved.

Reachability

The *reachability problem* is defined as follows: given an initial state of a hybrid automaton and a set of goal states, decide whether there exists a run starting in

the initial state that visits a state from the set of goal states. We assume that the controller has total control over the system.

Timed automata. Alur and Dill [AD94] proved that this problem is PSPACE-complete for timed automata. To show PSPACE membership they used a finite equivalence relation on the state space of the automaton — known as the region equivalence — which is exponential in the size of the automaton (constants are encoded in binary). Their proof of PSPACE-hardness used a reduction from the halting problem of a linear-space Turing machine with a word of length n on input, to the reachability problem of a timed automaton with $2n + 1$ clocks. Courcoubetis and Yannakakis [CY92] sharpened the result by showing that the problem is PSPACE-hard for timed automata with at least three clocks. Laroussinie et al. [LMS04] have shown that for single-clock timed automata the reachability problem is NLOGSPACE-complete. The complexity of the problem for two-clock timed automata remains open.

Linear hybrid automata. Alur et al. [ACHH93] have shown that the reachability problem is undecidable already for two-rate linear hybrid automata with three variables. A *two-rate linear automaton* is a linear hybrid automaton that is restricted in the following way: there are two rates and in every location the valuation of every variable changes at one of those two rates. The proof is by a reduction from the halting problem of two-counter machines. The value n of a counter is encoded as the value $\frac{1}{2^n}$ of a real-valued variable. The encoding of counter increments and decrements is facilitated by the fact that, in the definition of the jump predicate, the values of variables with different rates can be compared.

Alur et al. [ACH⁺95] show decidability of the reachability problem on *multi-rate linear hybrid automata* with the jump predicate restricted to boolean combinations of inequalities that involve a single variable and an integer constant. The result is obtained through a polynomial-time reduction to the reachability-problem

on timed-automata. Hence the problem is PSPACE-complete.

Rectangular hybrid automata. Puri and Varaiya [PV94] show that for a restricted subclass of rectangular hybrid automata the reachability problem is decidable. The restrictions are twofold: 1) the jump predicate is specified using boolean combinations of inequalities in which a value of the variable is compared against an integer constant; 2) if a discrete transition can occur at a non-integer time-point, the jump predicate guarantees reinitialisation of every variable whose flow changes.

Henzinger et al. [HKPP98] consider a different subclass of rectangular hybrid automata, referred to as the *initialised rectangular hybrid automata*, and show that the reachability problem for this class is PSPACE-complete. PSPACE-completeness follows from a polynomial-time reduction from an initialised rectangular hybrid automaton with n variables to a timed automaton with $2n + 1$ variables. The reduction consists of an intermediate reduction to a multi-rate linear automaton with $2n + 1$ variables. Every variable of the original initialised hybrid automaton is encoded using two variables of the multi-rate linear hybrid automaton. Intuitively, the two variables “keep track” of the upper and lower bounds on the value of the variable in the original initialised rectangular hybrid automaton. The two restrictions necessary for the result to hold are as follows: 1) values of two variables with different flows are never compared; 2) whenever the flow of a variable changes the value of that variable is reinitialised (reset). The authors also prove that if any of the two restrictions are removed then the reachability-problem becomes undecidable.

O-minimal hybrid automata. The reachability related results consist of establishing the existence of a special finite equivalence relation, referred to as a finite bisimulation, on the state space of the o-minimal hybrid automaton [LPS00; BM05; VPVD08]. The fact that such a finite bisimulation exists does not automatically imply decidability of the reachability problem because the construction of the postulated finite bisimulation is not necessarily effective [LPS00; BM05]. If a finite bisim-

ulation can be constructed the decidability of the reachability-problem is obtained through a reduction to the reachability problem on a finite graph. The effectiveness of the bisimulation construction depends on the decidability of the underlying o-minimal theory [LPS00; BMRT04; BM05; Gen05; VPVD08].

Lafferriere et al. [LPS00] prove that there exists a finite bisimulation for every o-minimal hybrid automaton. The result is obtained under the assumption that in every location and for every valuation of the real-variables, there is only one admissible trajectory. The crucial observation is that due to the restrictions placed on the jump predicate, one can decouple the discrete and continuous dynamics of an o-minimal hybrid automaton. The authors use this observation to construct a finite bisimulation for the continuous system in every location independently. It is the o-minimality property that facilitates the construction of a finite bisimulation.

Brihaye et al. [BMRT04; BM05] extend on the results of Lafferriere et al. by relaxing the assumption on uniqueness of trajectories — their construction allows for self intersecting trajectories. To construct the bisimulation they use the so-called “word encoding” technique. In this technique, given a finite partition of the set of all possible variable values, a trajectory is abstracted by a word, which is an ordered sequence of elements of the partition (in the order they are visited by the trajectory). The o-minimality property is used to prove that every trajectory is abstracted by a finite word, and that the set of all such abstractions is finite. The elements of this set of words are in fact used to construct the finite bisimulation.

Gentilini [Gen05] considered two extensions to the model of o-minimal hybrid automata and showed that in both cases finite bisimulations exist. Firstly, *relaxed o-minimal hybrid automata* are obtained by allowing the real-valued variables to retain their value upon a discrete transition, provided that in every cycle of control locations there is a transition that resets all of their values at the same time. Secondly, *MasterSlave o-minimal hybrid automata* are obtained by allowing one variable, referred to as the *master variable*, to retain its value upon a discrete

transition as long as its flow does not change. The flow of the master variable is independent of the other variables, referred to as the *slave variables*, which have to be reset on every discrete transition. Additionally, it is required that there exists an upper bound on the frequency of discrete transitions.

Vladimerou et al. [VPVD08] study the reachability problem on STORMED hybrid automata, an extension of o-minimal hybrid automata, and show how to construct a bisimulation for such an automaton.

Reachability games

In reachability-games the goal of player Min, who models the controller, is to reach a certain designated set of goal locations, and the goal of player Max, who models the adversarial environment, is to prevent that from happening. To solve the game means to compute the set of states from which player Min can win, and to synthesise a controller for reachability means to compute a strategy for player Min that ensures winning. The decision problem associated with reachability games is formulated as follows: given a state, decide if player Min can win from that state.

Timed automata. The controller synthesis problem for timed automata was first introduced by Hoffman and Wong-Toi [WTH92a; WTH92b]. They have established decidability of the controller synthesis problem for timed automata by adapting the approach of Ramadge and Wonham [RW89] to the region abstraction of timed automata. In their approach the players interact by independently choosing their moves, an actual transition of the automaton is determined by those choices.

Asarin et al. [AMP95] further studied the problem of controller synthesis for reachability and provided a symbolic algorithm. Their approach improved on the approach of the predecessors by avoiding the explicit construction of the region abstraction.

Henzinger and Kopke [HK99] established that the problem of synthesising

a controller for reachability on timed automata is EXPTIME-complete. Jurdziński and Trivedi [JT07] sharpened that result by showing the problem to be EXPTIME-complete for timed automata with at least two clocks.

De Alfaro et al. [dAFH⁺03] adopt a different approach to reachability games in which players may surprise each other with their actions. In their work, players interact in the following way: at every state, both players choose a time delay (a continuous transition) and an action (discrete transition) to perform once the requested time elapses. The system executes the choice (a pair consisting of a continuous and discrete transitions) with the shorter delay. A natural problem that arises is that players, to attain their objective, may attempt to stop time (Zeno behaviour), i.e., make their choices in such a way that infinitely many discrete transitions occur in finite time. Such behaviours are undesirable as they are not physically meaningful [dAFH⁺03]. The authors propose a method for eliminating such behaviours, and show that in this setting the problem of synthesising a controller for reachability is EXPTIME-complete. Eliminating Zeno-behaviour comes at the cost of non-uniform determinacy, i.e., from some states neither player can win.

Rectangular hybrid automata. Henzinger and Kopke [HK99] studied reachability games on rectangular hybrid automata. They considered a model in which the continuous transitions are under the control of the environment, and the discrete transitions are under the control of the controller. Additionally, the time is discrete, i.e., the controller can execute discrete transitions only at integer time points. They provide an algorithm for synthesising controllers for reachability and show that the problem is EXPTIME-complete.

Henzinger et al. [HHM99] consider controller synthesis for reachability in the context of *rectangular hybrid games*. The approach differs from the one presented by Henzinger and Kopke [HK99] in that it does not distinguish between the discrete controller and continuous environment — the environment and the controller

can both perform continuous and discrete transitions. In this setting, both players choose their transitions independently and one of them is nondeterministically chosen to be executed. The authors prove that controller synthesis for reachability in rectangular hybrid games is EXPTIME-complete. De Alfaro et al. [dAHM01] proposed a symbolic algorithm for synthesising controllers for reachability in rectangular hybrid games.

O-minimal hybrid automata. Bouyer et al. [BBC10; BBC06] study controller synthesis for reachability in the context of o-minimal hybrid automata. They consider an asymmetric setting in which the environment is the more powerful player. The approach bears similarities with the “element of surprise” approach introduced by de Alfaro [dAFH⁺03], however here the environment makes their choice knowing the choice of the controller, i.e., the controller can not surprise the environment. Under the assumption that the underlying o-minimal theory is decidable, Bouyer et al. prove that the controller synthesis problem for reachability on o-minimal hybrid automata is decidable, and provide an algorithm for synthesising such a controller. They employ the so-called “suffix word encoding” abstraction technique, which generalises the “word encoding” abstraction [BMRT04; BM05] to the game setting.

Vladimerou et al. [VPVD09] generalise the results of Bouyer et al. to the setting of reachability games on STORMED hybrid automata.

1.2.3 Quantitative analysis

In the previous section, we have surveyed problems with qualitative objectives; in this section we consider problems with quantitative objectives. Firstly, we discuss the reachability-price optimisation and average-price-per-reward ratio optimisation on hybrid automata. Secondly, we discuss controller synthesis for the aforementioned problems.

Optimal reachability-price

The *reachability-price problem* is defined on a weighted transition system, in which every transition bears a price: given an initial state, a set of goal states, and a constant c , determine whether there exists a run starting in the initial state that visits a state from the set of goal states, and whose accumulated price of transitions is at most c . A special case of this problem, referred to as *reachability-time problem*, occurs when the price of every discrete transitions is 0, and the price of every continuous transition is equal to its duration.

Timed automata. Weighted timed automata were introduced independently by Alur et al. [ALP01] and Behrmann et al. [BFH⁺01]. In this model every transition bears a price. The price of a continuous transition is equal to its duration multiplied by a control-location-specific integer constant, and the price of a discrete transition is equal to a control-action-specific integer constant. Such automata are referred to as *linearly-priced automata*.

The reachability-time problem for timed automata was shown to be decidable by Courcoubetis and Yannakakis [CY92], and Alur et al. [ACH97] have shown that the problem is PSPACE-complete.

Alur et al. [ATP04] proposed an EXPTIME algorithm for solving the reachability-price problem on linearly-priced timed automata. Their method used a non-trivial extension of the region abstraction.

When restricted to corner states, i.e., states in which all clock values are integer, Bouyer et al. [BBBR07] proved that the reachability-price problem for linearly-priced timed automata is PSPACE-complete. PSPACE-completeness is shown using the corner-point abstraction, a refinement of the region abstraction. The key observation behind the proof is that to achieve optimality for corner states, it is sufficient to consider only runs that go through (or very close to) corner states. PSPACE-hardness follows from the complexity of the reachability-problem for timed

automata [AD94]. For single-clock timed automata the reachability-price problem is NLOGSPACE-complete [LMS04].

Jurdziński and Trivedi [JT08b] generalise the results of Bouyer et al. and show that the reachability-price problem is PSPACE-complete for all initial states, not just the corner states. They use an abstraction technique, known as boundary-region abstraction, which generalises the corner-point abstraction.

O-minimal hybrid automata. Gentilini [Gen05] studied the reachability-time problem for o-minimal hybrid automata. She proved that the problem is decidable, provided that the underlying o-minimal theory is decidable. Brihaye et al. [BBC09; BBC07] extended this result to the reachability-price setting under the assumption that all prices are positive. In both works, the crucial observation is that it is sufficient to consider runs of bounded length. This is because the automaton is defined using an o-minimal theory, and because upon every discrete transition all of the real-valued variables are reinitialised.

Optimal price-per-reward

The *price-per-reward problem* is defined for doubly-weighted transition systems. Every transition bears two weights, a price and a reward, and we want to optimise the limit ratio of the accumulated price and reward in an infinite run. The price-per-reward problem is defined as follows: given an initial state and a constant c , determine whether there exists an infinite run whose limit ratio of the accumulated price and reward is at most c .

Timed automata. Bouyer et al. [BBL04] introduced the price-per-reward problem for doubly-weighted timed automata. In their considerations the automata were linearly-priced and linearly-rewarded. Using the aforementioned corner-point abstraction, they showed the problem to be PSPACE-complete for all corner states. The key observation is the same as in the context of the reachability-price prob-

lem [BBBR07], i.e., that it is sufficient to consider runs visiting only corner states or states that are sufficiently close to corner states.

Jurdziński and Trivedi [JT08b] extend the results of Bouyer et al. to all states of the automaton. They use a generalisation of the corner point-abstraction, known as the boundary-region abstraction, to show that the problem is PSPACE-complete.

Reachability-price games

Reachability-price games are played on weighted transition systems. The goal of player Min, who models the controller, is to assure that a goal state is reached, and if that is possible, to minimise the accumulated price of doing so. The goal of player Max, who models the adversarial environment, is to prevent player Min from reaching the goal state, and if that is not possible, to maximise the accumulated price of reaching a goal state. To solve the game means to compute the value of the game. To synthesise a controller means to compute a strategy for player Min that can ensure an almost optimal execution of the system. Finally, the decision problem associated with reachability-price games is as follows: given a state of the automaton and a constant c , decide if the value of the reachability-price game from that state is at most c .

Timed automata. Asarin and Maler [AM99] were the first to study reachability-time games on timed automata. They have proposed an algorithm which computes the uniform solution to the reachability-price game, i.e., it computes a function that given a state returns the upper value of a reachability-price game. They use a value iteration approach, which iteratively computes a solution to a certain set of equations whose solutions coincide with the upper values. The iterative procedure is guaranteed to terminate only on timed automata that are structurally non-Zeno, i.e., a sequence of transitions that results in a control cycle takes at least one unit of time.

Jurdziński and Trivedi [JT07] proved that reachability-time games are determined. Firstly, they characterise the game values using a set of equations, the so-called optimality equations. Secondly, they proposed a symbolic strategy improvement algorithm for computing the solution to these equations. In their approach, a strategy is a mapping from an element of the region abstraction to a function that maps each state in the region to an actual move of a player. In each iteration, the strategy improvement evaluates the strategies (assigns a value) and attempts to improve them (increase their value). The algorithm terminates when no improvement is possible. Jurdziński and Trivedi argue that the set of such symbolic strategies is finite for both players and that in each iteration the value of the strategy increases. Hence the strategy improvement procedure is guaranteed to terminate. Additionally, they establish that the reachability-price game problem is EXPTIME-complete for timed automata with at least two clocks.

Brihaye et al. [BHPR07] adopt the approach of games with an “element of surprise”, first introduced by [dAFH⁺03]. In the reachability-time game setting, as was the case with reachability games, players play by choosing a time delay and action to perform after the chosen time elapses; the action with a shorter time delay is executed. To avoid Zeno behaviours, the strategies available to players are restricted. It is known that in this restricted setting the game does not have to be determined [dAFH⁺03], however, Brihaye et al. provide an algorithm for computing the upper value of the game.

La Torre et al. [LTMM02] study reachability-price games on a restricted class of linearly-priced timed automata. They consider linearly-priced timed automata whose control graph does not contain cycles. In this restricted setting, they give a doubly-exponential algorithm for solving reachability-price games.

Reachability-price games on arbitrary linearly-priced timed automata were studied independently by Alur et al. [ABP04] and Bouyer et al. [BCFL04]. In both cases, the authors proposed a semi-algorithm for solving reachability-price games

on timed automata. These two algorithms are guaranteed to terminate on timed automata in which the price of every control cycle is strictly positive.

Alur et al. [ABP04] provide an algorithm for solving bounded reachability-price games, i.e., games in which the players are trying to optimise the price of reaching the goal state in a given finite number of steps. To solve this bounded variant of reachability-price games they use a refinement of the region abstraction. Additionally, they provide examples that show that the size of this refined abstraction grows exponentially with the number of steps.

Brihaye et al. [BBR05] show that deciding whether an optimal strategy exists in a reachability-price game on an automaton with at least five clocks is undecidable. The proof is by a reduction from the halting problem of a two-counter machine. Bouyer et al. [BBM06] sharpened this result by providing a reduction from the halting problem of a two-counter machine to a timed automaton with at least three clocks.

Bouyer et al. [BLMR06] studied reachability-price games on single-clock timed automata, and proposed a triply-exponential algorithm for solving them. They show that the problem can be reduced to a problem on a very simple subclass of single-clock timed automata and they provide an algorithm that solves the problem for this subclass. The algorithm is recursive with respect to the number of control locations in the automaton, and it is built around a very involved understanding of the structure of the solution. The exact complexity of the problem remains open.

O-minimal hybrid automata. Vladimerou et al. [VPVD09] study reachability-price games on STORMED hybrid automata, a generalisation of o-minimal hybrid automata. They propose a reduction from the reachability-price game decision problem on linearly-priced STORMED hybrid automata to the reachability game decision problem on STORMED hybrid automata. The reduction is based on the

following idea: provided that the STORMED hybrid automaton is linearly-priced, the price of a run can be encoded in an auxiliary variable, that is never reinitialised. The reachability-game decision problem is decidable for STORMED hybrid automata [VPVD09].

Average-price-per-reward games

In *average-price-per-reward games* the goal of player Min, who models the controller, is to minimise the limit ratio of the accumulated price and reward in an infinite run. The goal of player Max, who models the environment, is to maximise this ratio. To solve the game means to compute the value of the game, and to synthesise a controller means to compute a strategy that guarantees an almost optimal execution of the system. The associated decision problem is as follows: given a state and a constant c , decide whether the value of the average-price-per-reward game from that state is at most c .

Timed automata. Jurdziński and Trivedi [JT08a] consider average-time-per-transition games on timed-automata and prove that they are determined. They reduce the problem of computing the value of the game to the problem of computing the value of a finite average-price game. Additionally, they prove that the decision version of the problem is EXPTIME-complete for timed automata with at least two clocks.

Adler et al. [AdAF05] consider average-price-per-time games on timed automata with discrete time. The authors adopt the setting of games with an “element of surprise” [dAFH⁺03; BHPR07], i.e., players choose time delays and actions to execute after the chosen time elapses and the system executes the action which occurs first. However, in this case the time is discrete and players can choose to wait either 1 or 0 units of time only. Additionally, as it was the case with reachability games with an element of surprise [dAFH⁺03] and reachability-time games with an

element of surprise [BHPR07], players are forbidden to block time. The authors show that the games are not determined in this setting, and provide a $\text{NP} \cap \text{coNP}$ algorithm for computing the upper and lower values.

1.3 Contributions

In this thesis we consider reachability-price and price-per-reward games on three different models: *single-clock timed automata*, *price-reward game graphs*, and *hybrid systems with strong resets*. In each case, we prove that the games are positionally determined, and that almost optimal controllers can be synthesised. Below, we briefly discuss the contributions and the organisation of the thesis.

Single clock timed automata. In Chapter 3 we introduce single-clock timed automata and study turn-based reachability-price games on this model. The work presented in that Chapter extends the work of Bouyer et al. [BLMR06].

Firstly, we study *cost functions*, a class of piecewise-affine functions from bounded intervals to the set of reals augmented with the positive infinity. We prove several important properties of functions in this class that are instrumental in establishing the results regarding reachability-price games on single-clock timed automata. We believe that these results are interesting in their own right. Secondly, we show an EXPTIME algorithm for computing the value of a reachability-price game. As in Bouyer’s et al. [BLMR06] approach, the algorithm does the computation through a recursive procedure, with respect to the number of control locations of the timed automaton. Again, as in the work of Bouyer’s et al., in each recursive call we single out a control location that minimises the price rate. Control locations are assigned to players and handling of a control location by the algorithm depends on its ownership. In the case of control locations of player Max, our algorithm behaves exactly like the original, however, when it comes to handling control locations of player Min we improve over the predecessor. The original algorithm would proceed

to recursively solve two subproblems, which resulted in an additional exponential blowup. The algorithm presented in this thesis, as in the case of control locations of player Max, employs an iterative procedure which prevents this blowup. The approach is similar in spirit to that used in handling locations of player Max but the technical details are different.

Our contribution is threefold. Firstly, we provide an algorithm that improves the complexity of computing the value of reachability-price games on single-clock timed automata. We show the reachability-price game problem on single-clock timed automata is in EXPTIME, whereas the previously known best upper bound was 3EXPTIME [BLMR06]. Secondly, we provide results regarding cost functions, which are interesting in their own right, and are instrumental in establishing the correctness and complexity of the algorithm. Thirdly, we provide a rigorous analysis, previously absent in the literature, of the operations necessary to implement the algorithm.

This work was published in the paper titled “Two-Player Reachability-Price Games on Single-Clock Timed Automata” in the proceedings of the *Ninth Workshop on Quantitative Aspects of Programming Languages QAPL’11* [Rut11].

Price-reward game graphs. In Chapter 4, we introduce the notion of a price-reward game graph. Price-reward game graphs are an extension of doubly-weighted graphs (graphs in which every edge is assigned two weights: the price and the reward), where the price and the reward of an edge are dynamic, i.e., determined in the course of the game. In contrast to doubly-weighted graphs, a player’s move not only consists of choosing the next edge to traverse, but also of choosing edge-specific, inputs. This choice of inputs, made by both players, determines the price and reward of the traversed edge.

Firstly, we consider average-price-per-reward games, on infinite price-reward game graphs and for this model we formulate a special set of equations, the so-called optimality equations. We show that the solutions to these equations coincide

with game values. Secondly, we consider price-reward game graphs where the set of states and edges are finite. Note that there is no such restriction on the sets of edge inputs. In this setting we prove that average-price-per-reward games are determined and that almost optimal controllers can be synthesised.

To obtain the results, we first consider graphs of out-degree one, where players' actions consist of choosing edge inputs only. We obtain determinacy and synthesizability of almost-optimal controllers by showing that solutions to optimality equations, in this restricted setting, indeed exist. We use the technique of strategy improvement to lift this result first to a single-player setting and then to the two-player setting.

Notice that when both players choose their positional strategies, i.e., which edges to traverse from every state, they in fact choose a certain graph of out-degree one. This, and the result for graphs of out-degree one allow us to apply strategy improvement. Since the sets of positional strategies are finite, the algorithm terminates and the solution to the optimality equations in the final graph of out-degree one is the solution to the optimality equations for the whole game. It is worth noting that although in the actual game a player's strategy consists of choosing edges to traverse as well as choosing inputs for every traversed edge, the strategy improvement procedure operates only on the "edge choosing" components of players' strategies.

To the best of our knowledge price-reward game graphs have not been considered previously. Therefore, both the model and the results are novel. The results were first published by Marcin Jurdziński, Ranko Lazić and myself in the paper titled "Average-price-per-rewards games on hybrid automata with strong resets" in the proceedings of the *Tenth International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'09)* [JLR09], and later appeared in the special issue of the *International Journal on Software Tools for Technology Transfer* committed to the best papers submitted to the aforementioned conference [RLJ10].

Hybrid systems with strong resets. In Chapter 5 we introduce hybrid automata with strong resets and define the notion of a two-player zero-sum game on this model. In our definition of the game we adopt the approach of Bouyer et al. [BBC10; BBC06], i.e., we are considering an asymmetric setting in which player Max is the more powerful player. The approach bears similarities with the “element of surprise” approach introduced by de Alfaro [dAFH⁺03] however here, player Max makes their choice knowing the choice made by player Min, i.e., player Min can not surprise player Max.

We show that average-price-per-reward and reachability-price hybrid games with strong resets are positionally determined, and that positional almost optimal controllers can be synthesised. To obtain these results, we introduce an equivalence relation on the state space of the hybrid automaton that enables us to construct a respective “finite game” whose game values correspond to those of the original hybrid game. For both the hybrid and finite games we introduce a special set of equations, the so-called optimality equations, and prove that the solutions to those equations coincide with the game values. We use this optimality equation characterisation to prove that the values of the finite game coincide with the values of the original hybrid game.

In the asymmetric setting, the two players choose a transition in three steps. Every step is hybrid in nature, i.e., consists of choosing a discrete and a continuous component. Twice the choice consists of choosing an action (discrete) and a time (continuous), and once of choosing a state, which can be viewed as choosing an equivalence class (discrete) and then a state in that class (continuous). The aforementioned equivalence relation equates two states if they admit the same discrete interaction of both players. We use its equivalence classes as building blocks to create a finite graph in which the discrete choices made by both players, during their interaction, are encoded in the choices of edges to traverse. In case of average-price and reachability-price games we can disregard the continuous components, and the

“finite graph” we build is in fact a doubly-weighted finite graph. On the other hand, in case of average-price-per-reward games, we need to account for the continuous components, and the “finite graph” we build is in fact a price-reward game graph (as introduced in Chapter 4), where the continuous choices are encoded in the choices of the edge inputs.

Hybrid systems with strong resets were first introduced as o-minimal hybrid systems [LPS00; BM05]. The name stemmed from the fact that the flow, guard, and reset predicates were first-order definable over an o-minimal structure. The crucial property of the o-minimal structure was that every first-order definable subset of the domain was a finite union of intervals. This, together with the strong reset assumption, allowed for encoding the systems executions using words of finite length. This encoding was instrumental in establishing decidability of the reachability problem [LPS00; BM05], and in computing the upper value of reachability-price games [BBC07].

In our considerations we have restricted the structure, over which the sets are defined, to the field of reals. We chose this structure due to the existence of Tarski’s quantifier elimination procedure [Tar51], which facilitates the computational results. Although this structure is o-minimal, we chose to change the name of the model to hybrid automata with strong resets because the strong reset property is the property that is crucial to establishing our results. Determinacy and existence of almost optimal controllers hold regardless of the structure chosen.

Our contribution is threefold. Firstly, we consider average-price-per-reward games, which were not considered before, and we improve the reachability-price game results by removing the requirement that the price function is positive (as was required in [BBC07]). Secondly, we introduce a novel equivalence relation, which does not require o-minimality of the underlying structure. Thirdly, we extensively use optimality equations, which were not used in this context before.

The equivalence relation, and the results regarding average-price games and

reachability-price games were first published by Patricia Bouyer-Decitre, Thomas Brihaye, Marcin Jurdziński, Ranko Lazić, and myself in the paper titled “Average-price and reachability-price games on hybrid automata with strong resets” that appeared in the proceedings of the *Sixth International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS’08)* [BBJ⁺08]. The results regarding average-price-per-reward games were first published by Marcin Jurdziński, Ranko Lazić and myself in the paper titled “Average-price-per-rewards games on hybrid systems with strong resets” in the proceedings of the *Tenth International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI’09)* [JLR09], and later appeared in the special issue of the *International Journal on Software Tools for Technology Transfer* [RLJ10].

1.4 Organisation

The remainder of the thesis is organised as follows.

In Chapter 2 we provide a technical introduction. Firstly, we formally introduce the notion of definability, which will be used to finitely represent infinite structures (Section 2.1). Secondly, we introduce the notion of zero-sum games (Section 2.2). We consider two forms of representation for such games: strategic form and graph form (games on transitions systems). Two instances of zero-sum games on graphs are discussed in more detail: reachability-price and average-price-per-reward games — these are the games we will be considering in this thesis. In the conclusion of this chapter we formally introduce the notion of hybrid automata and their transition semantics (Section 2.3). Two classes of hybrid automata are discussed in more detail: timed automata and hybrid automata with strong resets — the two classes that we will focus on in this thesis. We also briefly explain how we define games on hybrid automata.

In Chapter 3 we study turn-based reachability-price games on single-clock

timed automata. We prove that the reachability-price game problem for this model is in EXPTIME. This result improves on the result of Bouyer et al. [BLMR06], which established a triply-exponential algorithm for this problem. The exact complexity of the problem remains open.

In Chapter 4 we introduce price-reward game graphs and study average-price-per-reward games on this model. Price-reward game graphs are an extension of doubly-weighted graphs, where the two weights of an edge, the price and the reward, are dynamic, i.e., determined by the choices made by both players. We show that average-price-per-reward games on finite price-reward game graphs are determined and that almost optimal controllers can be synthesised. In Chapter 5 the problem of solving average-price-per-reward games on hybrid automata with strong resets will be reduced to the analogous problem on finite price-reward game graphs.

In Chapter 5 we introduce hybrid automata with strong resets and define games on this model. Firstly, we introduce average-price-per-reward games and reachability-price games on hybrid automata with strong resets. Secondly, we prove that the problem of solving (synthesising controllers for) average-price-per-reward games and reachability-price games on hybrid systems with strong resets can be reduced to the analogous problem on finite price-reward game graphs and finite weighted graphs, respectively.

Chapter 2

Preliminaries

The purpose of this chapter is to provide a technical introduction and to put the results, presented in this thesis, in the broader context of the field. We introduce concepts such as definability, zero-sum games, and hybrid automata.

Firstly, we introduce means for representing infinite structures — the notion of logical definability. We also discuss how this concept can be used to obtain algorithmic results.

Secondly, we introduce zero-sum games, a variant of two-player games. Depending on the formalisation of the game play, we have different representations of zero-sum games. In Section 2.2.1 we formally introduce the concept of *zero-sum games in strategic form*, whereas, in Section 2.2.2, we introduce *zero-sum games on graphs*. To complete the introduction to zero-sum games on graphs, we discuss two instances of zero-sum games on graphs, namely *average-price-per-reward* and *reachability-price* games.

Thirdly, in Section 2.3 we formally define hybrid automata and their transition semantics, and then proceed to discuss two of their subclasses, of interest to this thesis: *hybrid automata with strong resets* (Section 2.3.1) and *timed automata* (Section 2.3.2). The first subclass is characterised by complex continuous transitions, at the cost of restrictions placed on the discrete ones. The second subclass,

on the other hand, places little restrictions on discrete transitions, but the continuous transitions are limited to changes to the variable values that occur at a constant and uniform rate. Finally, in Section 2.3.3 we explain how to define zero-sum games on hybrid automata using the notion of a game on a transition system.

2.1 Definability and decidability

In this thesis the algorithmic results concern models that have potentially infinite structures. In order to formulate those results we need to have a framework for representing finitely and reasoning about the infinite structures involved. For that purpose we are going to use the notion of *logical definability*. To be more precise, to represent infinite sets, we are going to use first-order formulae over $\mathcal{M} = \langle \mathbb{R}, 0, 1, +, \cdot, \leq \rangle$, the ordered field of reals (A general reference for first-order logic is the work of Hodges [Hod97]).

Example 2.1. *The intersection of the set $[1, 2]^2$ with the set of points satisfying the equation $y = x^2$ can be defined using the following formula:*

$$\Phi(x, y) = (x \leq 2) \wedge (x \geq 1) \wedge (y \leq 2) \wedge (y \geq 1) \wedge (y = x \cdot x),$$

where 2 is a shorthand for $1 + 1$, formula $a = b$ can be expressed as $(a \leq b) \wedge (b \leq a)$, and the formula $a \geq b$ can be expressed as $\neg(a \leq b) \vee (a = b)$. \square

Definition 2.2 (Definibility). *A set $X \subseteq \mathbb{R}^n$ is said to be first-order definable (for short, definable) iff there exists a first-order formula $\phi(z)$, with one free variable z , over the structure $\mathcal{M} = \langle \mathbb{R}, 0, 1, +, \cdot, \leq \rangle$ such that $x \in X$ iff $\mathcal{M} \models \phi(x)$.*

The first-order theory of \mathcal{M} is the set of all first-order sentences that are true in \mathcal{M} . We chose structure \mathcal{M} because, by Tarski's well-known result [Tar51], its first-order theory is decidable.

Using definability we can finitely represent sets and relations on real numbers and hence also real functions. We argue that we can approximate the values of such functions on rational arguments. We restrict our considerations only to rational arguments, as they have an obvious finite representation.

Let A be a finite set. We say that $(a, x) \in A \times \mathbb{R}^n$ is *rational* if x is rational. This reflects the hybrid nature of the models considered. For instance, a transition in a hybrid automaton consists of two components: a label drawn from a finite set, and a duration, which is a real number.

Let us consider a partial function $f : X \rightarrow \mathbb{R}$ which is defined on a set $D \subseteq X \subseteq A \times \mathbb{R}^n$. We can now introduce two key definitions that deal with computational issues.

Definition 2.3 (Decidable value). *Function $f : X \rightarrow \mathbb{R}$ is said to have decidable value if the following problem is decidable: given rationals $x \in D$ and a $c \in \mathbb{Q}$, decide whether $f(x) \leq c$.*

Definition 2.4 (Approximate computability). *Function $f : X \rightarrow \mathbb{R}$ is said to be approximately computable if there exists an algorithm that, for every rational $x \in D$ and every rational $\varepsilon > 0$, computes a $y \in \mathbb{Q}$, such that $|y - f(x)| < \varepsilon$.*

The following three propositions establish a relation between definability and computability.

Proposition 2.5. *If a real partial function is definable in \mathcal{M} , then it has decidable value.*

Proof. Let $f : X \rightarrow \mathbb{R}$ be a partial function which is defined on $D \subseteq X$. If f is definable then, for every rational $x \in D$ and rational c , $f(x) \leq c$ is a first-order sentence in \mathcal{M} , hence by the result of Tarski [Tar51], we can decide whether it is true, which leads to decidability of f . \square

Proposition 2.6. *If a function has decidable value then it is approximately computable.*

Proof. If a function has decidable value, we can identify its over and under approximations. Using binary search, we can get arbitrarily close approximations. \square

Proposition 2.7. *If a set X is definable and contains a rational element x , then some rational element of X can be computed.*

Proof. If a set $X \subseteq A \times \mathbb{R}^n$ is definable, then its characteristic function is definable, and decidable. The latter follows from Proposition 2.5.

To compute an element of X , it is sufficient to enumerate all rational elements, and test for containment — decide whether the value of the characteristic function, of X , is less or equal $1/2$. Since the set contains a rational element, the procedure has to terminate. \square

The purpose of the above definitions and results is to enable us to state conclusions of our definability results. They should not be treated as a formalisation of computation over the reals. For models of computing over the reals, we refer the reader to [Wei00; MM97; BSS89].

2.2 Zero-sum games

In this section we introduce the concept of zero-sum games. In a zero-sum game there are two players: Min and Max. The zero-sum name comes from the fact that the losses of the first player are the second player's gains, i.e., the sum of both players' winnings is always zero. The games discussed in this thesis, i.e., reachability-price and average-price-per-reward games, are zero-sum.

Zero-sum games admit different representations. We start this section by discussing the strategic-form representation. We formulate the fundamental concepts such as value of a game, determinacy, and optimal strategies. Further on in the section, we introduce another representations of zero-sum games, games on transition systems (see Section 2.2.2). We lift the concepts of game value, determinacy

and optimal strategies to this new setting. Additionally, in this context we discuss the notions of optimality equations and strategy improvement algorithms, which are instrumental to obtaining the results discussed in this thesis. We conclude the introduction to games transition systems by focusing on average-price-per-reward and reachability-price games.

Note that although the results presented in this thesis concern games on hybrid automata, the notion of a game in strategic form will be important throughout Chapters 3–5.

2.2.1 Games in strategic form

There are various ways of representing zero-sum games. One of them is to specify the set of available strategies for each player. A payoff function takes as arguments two strategies, one for each player, and returns the value of player Max's *payoff*, i.e., his winnings; by negating this value we get the payoff of player Min. This form of representation is called the strategic form.

Definition 2.8 (Strategic form). *A zero-sum game in strategic form is given by $\mathcal{G} = \langle \Sigma^{\text{Min}}, \Sigma^{\text{Max}}, P \rangle$, where:*

- $\Sigma^{\text{Min}}, \Sigma^{\text{Max}}$ are the sets of strategies for players Min and Max respectively,
- $P : \Sigma^{\text{Min}} \times \Sigma^{\text{Max}} \rightarrow \mathbb{R}$ is the payoff function, i.e. the payoff function of player Max.

In such a game, the players choose their strategies simultaneously and independently. Based on their choices, the payoff function P assigns the payoff. The objective of player Min is to minimise that value, whereas that of player Max is to maximise it.

As discussed in Section 2.1 we will use the concept of definability to represent potentially infinite objects. We say that \mathcal{G} is *definable* if all its components are

definable. Recall that definability of a component implicitly implies that it is a subset of \mathbb{R}^n .

A fundamental concept for zero-sum games is that of *game value*. Intuitively, this is the payoff that both players can guarantee, i.e., player Max can guarantee that the payoff will be no less than the game value, and player Min can guarantee that the payoff will not exceed the game value. Not all games have game values. If a game has a game value then we say that it is *determined*.

To define the concept of game value formally, we need to introduce some auxiliary concepts, such as best response, and lower and upper values.

Consider a situation in which player Max declares the strategy that he is going to play. This reduces the game to a minimisation problem, namely that of choosing an optimal¹ strategy for player Min. Such a strategy (of player Min) will be referred to as a best response. The value of player Max's strategy is that of the payoff when Max plays that strategy, and Min plays a best response. Formally, given a game \mathcal{G} , the value of strategy $\chi \in \Sigma^{\text{Max}}$ is defined as follows:

$$\text{Val}_\chi(\mathcal{G}) = \inf_{\mu \in \Sigma^{\text{Min}}} P(\mu, \chi).$$

Obviously, player Max is interested in declaring a strategy $\chi \in \Sigma^{\text{Max}}$ that maximises the strategy value. The value of Max's optimal strategy is referred to as the *lower value*:

$$\text{Val}_*(\mathcal{G}) = \sup_{\chi \in \Sigma^{\text{Max}}} \text{Val}_\chi(\mathcal{G}) = \sup_{\chi \in \Sigma^{\text{Max}}} \inf_{\mu \in \Sigma^{\text{Min}}} P(\mu, \chi).$$

We can interpret the lower value as the guaranteed payoff for player Max, assuming his rational behaviour.

Similarly, if player Min declares her strategy, we define the value of that

¹ In general such strategies do not exist. However, for now we assume that they do exist — this will aid the clarity of exposition. We will deal with this problem later in this chapter, using the concept of ϵ -optimality.

strategy in terms of player Max's best response. For a game \mathcal{G} and a strategy $\mu \in \Sigma^{\text{Min}}$ we define:

$$\text{Val}^\mu(\mathcal{G}) = \sup_{\chi \in \Sigma^{\text{Max}}} P(\mu, \chi).$$

The counterpart of the lower value is called the *upper value*:

$$\text{Val}^*(\mathcal{G}) = \inf_{\mu \in \Sigma^{\text{Min}}} \text{Val}^\mu(\mathcal{G}) = \inf_{\mu \in \Sigma^{\text{Min}}} \sup_{\chi \in \Sigma^{\text{Max}}} P(\mu, \chi).$$

Definition 2.9 (Determinacy). *A game \mathcal{G} in strategic form is said to be determined if*

$$\text{Val}_*(\mathcal{G}) = \sup_{\chi \in \Sigma^{\text{Max}}} \inf_{\mu \in \Sigma^{\text{Min}}} P(\mu, \chi) = \inf_{\mu \in \Sigma^{\text{Min}}} \sup_{\chi \in \Sigma^{\text{Max}}} P(\mu, \chi) = \text{Val}^*(\mathcal{G}).$$

For a determined game we write $\text{Val}(\mathcal{G})$ to denote the value of the game, which is equal to both its lower and upper values.

Note that $\text{Val}_*(\mathcal{G}) \leq \text{Val}^*(\mathcal{G})$ holds for every game \mathcal{G} , so to prove determinacy one needs to prove the opposite inequality only.

When introducing the lower and upper values, we have assumed the existence of optimal strategies. This is not always the case; there are two possible reasons: strategy values diverge to infinity, or strategy values converge but the limit value is never attained. The games we are considering in this thesis have bounded payoff functions, hence we need to handle only the latter reason for the non-existence of optimal strategies. We deal with this problem through the concept of ε -optimality.

Definition 2.10 (ε -optimality). *For $\varepsilon > 0$, we say that $\mu \in \Sigma^{\text{Min}}$ is ε -optimal if we have that:*

$$\text{Val}^\mu(\mathcal{G}) = \sup_{\chi \in \Sigma^{\text{Max}}} P(\mu, \chi) \leq \inf_{\mu \in \Sigma^{\text{Min}}} \sup_{\chi \in \Sigma^{\text{Max}}} P(\mu, \chi) + \varepsilon = \text{Val}^*(\mathcal{G}) + \varepsilon.$$

We define ε -optimality of Max's strategies analogously, that is $\chi \in \Sigma^{\text{Max}}$ is said to be ε -optimal if $\text{Val}_\chi(\mathcal{G}) \geq \text{Val}_(\mathcal{G}) - \varepsilon$.*

Remark 2.11. *There are cases in which the desired payoff function is only partially defined. To remedy this, a lower*

$$P_* : \Sigma^{\text{Min}} \times \Sigma^{\text{Max}} \rightarrow \mathbb{R}$$

and an upper

$$P^* : \Sigma^{\text{Min}} \times \Sigma^{\text{Max}} \rightarrow \mathbb{R}$$

payoff functions are used. It is required that $P_ \leq P^*$. With this generalisation, the lower (resp. upper) value, and the value of player Max's (resp. Min's) strategy are defined using the lower (resp. upper) payoff.* \square

2.2.2 Games on graphs

We will be considering turn-based games on doubly-weighted labelled transition systems. A labelled transition system is a (possibly infinite) graph with labels assigned to edges. In this context, vertices are referred to as states, edges are referred to as transitions, and the set of edges is referred to as the labelled transition relation. The two weight functions assign real numbers to every transition, and are referred to as the price and reward functions.

To define turn-based games on labelled transition systems, we need a notion of a game graph. We obtain a game graph from a doubly-weighted transition system by introducing a partitioning of its set of states between the two players, Min and Max. Intuitively the game is played by moving a token along the transitions, from one state to another. Every move incurs a price and a reward. The owner of the state decides along which transition to move the token.

A game graph can be seen as an alternative to the strategic form representation of a zero-sum game. In this context a payoff function determines how the prices (rewards) of individual moves contribute to the overall value of a particular play.

Transition systems. We will first introduce a doubly-weighted labelled transition system, then we will explain how to define a game graph, and finally we will define the notion of a game.

Definition 2.12 (Doubly-weighted labeled transition system). *A doubly-weighted labelled transition system, or simply a transition system, $\mathcal{T} = \langle \mathbf{S}, \Lambda, \rightarrow, \pi, \rho \rangle$ consists of the following components:*

State space. *A (possibly infinite) set \mathbf{S} .*

Transition relation. *A (possibly infinite) set of transition labels Λ and a ternary transition relation $\rightarrow \subseteq \mathbf{S} \times \Lambda \times \mathbf{S}$. An element of \rightarrow will be referred to as a transition, and we will write $s \xrightarrow{\lambda} s'$ to denote the transition $(s, \lambda, s') \in \rightarrow$. Note that \rightarrow induces a binary relation $\xrightarrow{\lambda} \subseteq \mathbf{S} \times \mathbf{S}$ for every label $\lambda \in \Lambda$.*

Weight functions *Two weight functions $\pi, \rho : \mathbf{S} \times \Lambda \times \mathbf{S} \rightarrow \mathbb{R}$ that assign a real number to every transition. The functions π and ρ are referred to as the price and reward functions.*

We say that the transition system \mathcal{T} is deterministic if, for every $\lambda \in \Lambda$, the binary relation $\xrightarrow{\lambda}$ is in fact a partial function $\xrightarrow{\lambda} : \mathbf{S} \rightarrow \mathbf{S}$.

Remark 2.13. *In reachability-price games the reward information is not taken into account, and hence the ρ function will be omitted from the specification in this context.* □

We now briefly recall the standard notions related to transition systems and their executions, also called as runs. Given a state s , a run of the transition system from s is a (possibly infinite) sequence of transitions $\omega = s_0 \xrightarrow{\lambda_1} s_1 \xrightarrow{\lambda_2} s_2 \dots$, where $s = s_0$. If two runs $\omega = s_0 \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_k} s_k$ and $\omega' = s'_0 \xrightarrow{\lambda'_1} \dots$ are such that $s_k = s'_0$ then, $\omega\omega'$ denotes the run $s_0 \xrightarrow{\lambda_1} \dots \xrightarrow{\lambda_k} s_k \xrightarrow{\lambda'_1} \dots$. Given a finite run ω , $\text{Len}(\omega)$ will denote its length, i.e., the total number of transitions, $\text{Last}(\omega)$ will denote the final state of ω , i.e., $s_{\text{Len}(\omega)}$, and ω_n will denote the prefix of ω of length

n , where $n \leq \text{Len}(\omega)$. The set of all runs of \mathcal{T} is denoted by Runs . The set of all finite runs of \mathcal{T} is denoted by Runs_{fin} . Note that $\text{Runs}_{\text{fin}} \subseteq \text{Runs}$. We will also write $\text{Runs}(s)$ ($\text{Runs}_{\text{fin}}(s)$) to denote the set of all runs (all finite runs) starting in a state s . Whenever the transition system \mathcal{T} is not clear from the context, we will use a superscript \mathcal{T} , e.g., we will write $\text{Runs}_{\text{fin}}^{\mathcal{T}}$ to denote the set of finite runs of the transition system \mathcal{T} .

Given a finite run $\omega \in \text{Runs}_{\text{fin}}$, its price is the accumulated price of all the transitions, i.e.:

$$\text{Price}(\omega) = \sum_{i=1}^{\text{Len}(\omega)} \pi(\lambda_i),$$

and its reward is the accumulated reward of all of its transitions, i.e.:

$$\text{Reward}(\omega) = \sum_{i=1}^{\text{Len}(\omega)} \rho(\lambda_i).$$

Games. We will now define games on doubly-weighted labelled transition systems, together with the standard concepts such as strategies and game value. In order to define the notion of a zero-sum game on a weighted labelled transition system we need to define a game graph, which is obtained by partitioning of the state space between the two players, Min and Max. The following definition formalises this concept.

Definition 2.14 (Doubly-weighted game graph). *A doubly-weighted game-graph on a transition system $\Gamma = \langle \mathcal{T}, \mathcal{S}^{\text{Min}}, \mathcal{S}^{\text{Max}} \rangle$ consists of*

- *a transition system \mathcal{T} , and*
- *a partition of its state space $\mathcal{S} = \mathcal{S}^{\text{Min}} \cup \mathcal{S}^{\text{Max}}$.*

A particular game is defined when a payoff function P is provided. The following definition formalises the concept of a game.

Definition 2.15 (Doubly-weighted game). *A doubly-weighted game is a pair $\langle \Gamma, P \rangle$ where:*

- Γ is a doubly-weighted game graph, and
- $P : \text{Runs} \rightarrow \mathbb{R}$ is a payoff function.

We will abuse notation and when the payoff function is clear from the context we will write Γ to denote the game. In this context, a transition $s \xrightarrow{\lambda} s'$ will often be referred to as a *move*.

Assumption 2.16. *When considering games on transition systems, we will be considering deterministic weighted labelled transition systems.* \square

The results regarding games on finite transition systems presented in this section will refer to deterministic transition systems. Games on doubly-weighted transition systems will be used to define games on single-clock timed automata (introduced in Chapter 3), which yield deterministic transition systems. Hybrid automata with strong resets (introduced in Chapter 5) yield transition systems that are potentially non-deterministic, but in that case the games are defined differently without referring to the concepts presented here. In this context, the restriction made in Assumption 2.16 does not apply. We place this restriction because it allows for simpler definitions, e.g., we rely on this restriction when defining the notion of a run induced by the players' strategies.

We now introduce the standard game-related definitions. A strategy of player Min is a partial function $\mu : \text{Runs}_{\text{fin}} \rightarrow \Lambda$ such that for every finite run ω ending in a state of the player Min, $\text{Last}(\omega) \xrightarrow{\mu(\omega)} s'$. We say that μ , a strategy of player Min, is positional if for every two runs $\omega, \omega' \in \text{Runs}_{\text{fin}}$, if ω and ω' end in the same state then $\mu(\omega) = \mu(\omega')$, i.e., it can be viewed as a function $\mu : S \rightarrow \Lambda$. We will write Σ^{Min} for the set of all strategies of player Min, and Π^{Min} for the subset of all of her positional strategies. The sets of strategies of player Max is defined analogously.

Given a run ω' ending in a state s_0 , and a pair of strategies $\sigma \in \Sigma^{\text{Min}}$ and $\chi \in \Sigma^{\text{Max}}$, we write $\text{Run}(\omega', \mu, \chi)$ to denote the unique run $\omega \in \text{Runs}(s_0)$ satisfying: if $s_i \xrightarrow{\lambda_{i+1}} s_{i+1}$ is the $(i+1)$ -th transition (recall that we are considering deterministic

transition systems only) of ω then $\mu(\omega'\omega_i) = \lambda_i$ if $s_i \in S^{\text{Min}}$, otherwise, $\chi(\omega'\omega_i) = \lambda_i$. Note that if μ and χ are positional then $\text{Run}(\omega', \mu, \chi) = \text{Run}(\text{Last}(\omega'), \mu, \chi)$.

Determinacy. We conclude by introducing the concept of game value. Firstly, we introduce the notion of the value of a game from a state. We do this by defining a zero-sum game in strategic form (see Section 2.2.1), which is specific for the given state. Secondly, we introduce the notion of game value as a function that assigns the value of the state-specific game to every state. Note that the game value is in fact a partial function from states to reals.

Given a state s , let $\mathcal{D}_s = \langle \Sigma^{\text{Min}}, \Sigma^{\text{Max}}, \mathbf{P}(\text{Run}(s, \cdot, \cdot)) \rangle$ be a zero-sum game in strategic form. We say that the game Γ is *determined* for state s if \mathcal{D}_s is determined, i.e., $\text{Val}_*(\mathcal{D}_s) = \text{Val}^*(\mathcal{D}_s)$, and positionally determined if:

$$\text{Val}(\mathcal{D}_s) = \inf_{\mu \in \Pi^{\text{Min}}} \text{Val}^\mu(\mathcal{D}_s) = \sup_{\chi \in \Pi^{\text{Max}}} \text{Val}^\chi(\mathcal{D}_s).$$

We say that a game Γ is determined if it is determined from every state. For simplicity, we will write $\text{Val}(s)$ rather than $\text{Val}(\mathcal{D}_s)$ in this context. Hence, Val can be viewed as a partial function $S \rightarrow \mathbb{R}$.

Definition 2.17 (Game value decidability). *We will say that a game Γ is decidable if the partial function $\text{Val} : S \rightarrow \mathbb{R}$ has decidable value.*

In the definition above we have emphasised that Val is a partial function because Γ does not have to be determined from every state.

Games on finite graphs

We will now focus on games on finite doubly-weighted game graphs. A finite doubly-weighted game graph is a game graph of a finite doubly-weighted labelled transition system with the property that every transition label uniquely determines the transition. This means that, given a state, choosing a transition amounts to choosing

the next state. Such transition systems are clearly deterministic. In this context we will be using the term “edge” rather than “transition”.

We start by introducing the definition of a finite doubly-weighted game graph, and showing how this definition relates to that of a game graph on a doubly-weighted labelled transition system. Then we proceed to discuss two techniques used in the context of games on finite doubly-weighted game graphs, which are relevant to this thesis.

Definition 2.18 (Finite doubly-weighted game graph). *A finite double-weighted game graph, or simply a finite game graph is given by $\Gamma = \langle S^{\text{Min}}, S^{\text{Max}}, E, \pi, \rho \rangle$, where:*

- $S^{\text{Min}} \cap S^{\text{Max}} = \emptyset$ are the sets of states for players Min and Max respectively,
- $\langle S^{\text{Min}} \cup S^{\text{Max}}, E \rangle$ is a finite directed graph,
- $\pi : E \rightarrow \mathbb{R}$ is the price function,
- $\rho : E \rightarrow \mathbb{R}$ is the reward function.

Remark 2.19. *If Γ was to be viewed as a doubly-weighted labelled transition system, then the transition relation could be defined as $s \xrightarrow{(s,s')} s'$ for every $(s, s') \in E$. \square*

The notion of a run is similar to the notion of a run of a doubly-weighted transition system, with the only difference that there are no labels (they are omitted as they are redundant). We write $s \rightarrow s'$ to denote a move, where $e = (s, s') \in E$. The price of the move is $\pi(e)$ and the reward is $\rho(e)$. A run is a (possibly infinite) sequence of moves $\omega = s_0 \rightarrow s_1 \rightarrow s_2 \dots$.

The definitions of players’ strategies differ slightly. In this context a strategy, rather than choosing a label of the next transition, chooses the next edge (transition) to traverse. Thus, a strategy of player Min is a function $\mu : \text{Runs}_{\text{fin}} \rightarrow E$, that is defined for all finite runs ending in a state $s \in S^{\text{Min}}$. A strategy of player Max is defined similarly. The notions of positional strategies and runs induced by strategies

naturally follow. Note that this definition is in fact equivalent to the original, but better reflects the syntax of finite doubly-weighted game graphs (Remark 2.19)

Optimality equations. We want to capture global optimality of player's strategies using local conditions. For that purpose, we introduce the notion of *optimality equations* [Bel57; Ber01], sometimes referred to as *Bellman equations*. A solution to these equations can be seen as a witness for existence of positional optimal strategies for both players.

Optimality equations will be discussed in more detail, later in this section, in the context of finite average-price-per-reward games and finite reachability-price games. Below, we present only the basic idea behind the concept. Moreover, we explain how optimality equations are being used to obtain the results presented in this thesis.

We say that a function $f : \mathcal{S} \rightarrow \mathbb{R}$ is a solution to the optimality equations for a game Γ , denoted by $f \models \text{Opt}(\Gamma)$, if the following equations are satisfied:

$$f(s) = \min_{(s,s') \in E} \text{expr}(s, s', f(s')),$$

for every state $s \in \mathcal{S}^{\text{Min}}$, and:

$$f(s) = \max_{(s,s') \in E} \text{expr}(s, s', f(s')),$$

for every state $s \in \mathcal{S}^{\text{Max}}$, where $\text{expr}(s, s', f(s'))$ is an algebraic expression involving s , s' , and $f(s')$.

The actual form of $\text{expr}(s, s', f(s'))$ depends on the game Γ , and needs to be chosen in such a way that we can formulate and prove a theorem of the following form:

Given a finite game Γ , if a function $f : \mathcal{S} \rightarrow \mathbb{R}$ is such that $f \models \text{Opt}(\Gamma)$ then $\text{Val}(s) = f(s)$ for every $s \in \mathcal{S}$.

The proof of the theorem is typically established by proving the following two inequalities:

$$\begin{aligned} f(s) &\geq \text{Val}^*(s) - \varepsilon, \\ f(s) &\leq \text{Val}_*(s) + \varepsilon, \end{aligned}$$

for every $s \in \mathbf{S}$ and $\varepsilon > 0$. To prove the first inequality we construct a positional strategy μ_ε for player Min in such a way that for every counter strategy χ we have $P(\text{Run}(s, \mu_\varepsilon, \chi)) \leq f(s) + \varepsilon$ for every $s \in \mathbf{S}$. The construction is based on the following principle: in every state $s \in \mathbf{S}^{\text{Min}}$ player Min can choose a state s' such that:

$$f(s) \geq \text{expr}(s, s', f(s')) - \varepsilon,$$

and if $s \in \mathbf{S}^{\text{Max}}$, the above inequality holds for all states s' , such that $(s, s') \in \mathbf{E}$. Similarly, the second inequality is proved by constructing an appropriate strategy for player Max.

The strategy constructed in the proofs of the inequalities is an ε -optimal positional strategy for a given player. Notice that a solution to the optimality equations establishes local optimality constraints, and that the strategy is constructed in such a way that these constraints are being satisfied. In this sense, the solution to the optimality equations can be seen as a witness for the existence of ε -optimal positional strategies for both players.

To summarise, optimality equations will be used as a characterisation of game values, and their solutions as a witness for existence of ε -optimal positional strategies. In Chapter 4 we will use this characterisation to prove that certain games are indeed determined, and in Chapter 5 we will use it to show that a certain reduction is correct. In the latter case, we will be dealing with a complex and a simple game. In both cases we will establish an optimality equation characterisation of game values, and then prove that the solutions to those two sets of optimality

equations are related to one another.

Strategy improvement. In the previous paragraph we have introduced the concept of optimality equations, and explained how to prove that if a solution to those equations exists then the game is determined and both players have ε -optimal positional strategies. We will use the concept of *strategy improvement* to prove that the solutions indeed exist. Strategy improvement is a technique that is used to compute ε -optimal positional strategies by applying local improvements.

We introduce the basic idea behind strategy improvement and its connection to optimality equations. Furthermore, we explain how it will be used in this thesis. Strategy improvement will be discussed in more detail in Chapter 4, in the context of average-price-per-reward games.

The algorithm relies on two assumptions. First, that in a single player game we can compute an positional optimal strategy for the distinguished player, i.e., given a positional strategy of one player, we can compute a positional best response strategy for the other player. Second, that when two positional strategies are fixed, and hence we are dealing with a zero-player game, a solution to the optimality equations exists.

Given a finite game Γ , a strategy improvement algorithm proceeds according to the following scheme:

1. Arbitrarily choose a strategy $\mu \in \Pi^{\text{Min}}$.
2. Compute the best response strategy $\chi \in \Pi^{\text{Max}}$.
3. Compute a solution f to the optimality equations $\text{Opt}(\Gamma_{\mu\chi})$. Positional strategies μ and χ induce a sub-game $\Gamma_{\mu\chi}$, a game in which every state has only one successor, i.e., the state that is chosen by the relevant strategy.

4. If for some $s \in S^{\text{Min}}$ there exists s' such that $(s, s') \in E$ and

$$f(s) > \text{expr}(s, s', f(s')),$$

then set $\mu(s) = s'$ and go to Step 2, otherwise terminate and output the functions f , μ , and χ .

Upon termination, the algorithm outputs the solution to the optimality equations $\text{Opt}(\Gamma)$, i.e., the function f and the ε -optimal positional strategies μ and χ for players Min and Max, respectively.

In Step 4, we are myopically improving the strategy of player Min. We are using the solution to the optimality equations for the zero-player game as an indicator of where the improvement should be made. The strategy χ is a best response strategy of player Max to μ , a strategy of player Min. Therefore, the solution f to $\text{Opt}(\Gamma_{\mu\chi})$ is such that for every $s \in S^{\text{Max}}$, we have $f(s) \geq \text{expr}(s, \chi(s), f(\chi(s))) - \varepsilon$. If this was not true, we would have a contradiction to the fact that χ is a best response, since f is a witness to the existence of an ε -optimal positional strategy.

If no improvement is possible then μ , and hence χ , are ε -optimal. Moreover, they are exactly the kind of strategies that would have been construed in the proof of the theorem that establishes that optimality equations characterise the game values, as discussed in the previous paragraph.

Given a strategy μ of player Min and a strategy χ , the best response of player Max, the solution to $\text{Opt}(\Gamma_{\mu\chi})$ can be seen as a valuation of strategy μ in every state. The sets of positional strategies for both players are finite. To prove termination of the strategy improvement algorithm it suffices to prove that after every iteration of the strategy improvement algorithm the new strategy of player Min in every state has a valuation that is not higher than the valuation of the previous strategy. Formally, to establish termination and correctness of a strategy improvement algorithm, it suffices to prove a theorem of the following type:

Let μ be a positional strategy of player Min, let χ be a positional best response strategy of player Max, and let f be a solution to the optimality equations for the zero-player game induced by those strategies. If μ' is an improvement of μ , as described in Step 4, χ' is the respective best response, and f' is the respective solution to the optimality equations, then the following holds for all $s \in S$:

$$f'(s) \leq f(s)$$

and the inequality is strict for some $s \in S$.

In Chapter 4 strategy improvement, together with optimality equations, will be used to prove determinacy of average-price-per-reward games on an extension of the finite doubly-weighted game graphs.

Average-price-per-reward games

In this section we introduce average-price-per-reward games on doubly-weighted labelled transition systems. We will focus on a variant of those games, referred to as finite average-price-per-reward games, which are played on finite game graphs. We will provide a characterisation of game values using a set of equations, referred to as average-price-per-reward optimality equations. The aim of this section is to provide the intuition for average-price-per-reward games, and why average-price-per-reward optimality equations indeed characterise game values. This intuition will be helpful when presenting the results in Chapters 4–5.

An *average-price-per-reward game* is played on a game graph Γ . In the remainder of this section we will abuse the notation, as explained earlier, and simply write Γ without specifying the payoff functions, to denote the average-price-per-reward game.

The goal of player Min in an average-price-per-reward game Γ is to minimise

the average price-over-reward ratio in a run, and the goal of player Max is to maximise it. As explained in Chapter 1, average-price-per-reward games are used in the context of recurrent-behaviour optimisation.

We define the upper and lower payoff functions in the following way:

$$\mathcal{A}^*(\omega) = \limsup_{n \rightarrow \infty} \frac{\text{Price}(\omega_n)}{\text{Reward}(\omega_n)}$$

and

$$\mathcal{A}_*(\omega) = \liminf_{n \rightarrow \infty} \frac{\text{Price}(\omega_n)}{\text{Reward}(\omega_n)}$$

Note that $\mathcal{A}_* \leq \mathcal{A}^*$, as required in Remark 2.11, where the concept of upper and lower payoffs is discussed.

Remark 2.20. *For the lower and upper average-price-per-reward payoffs to be well defined, the reward function has to be such that for every infinite run, only finitely many of its finite prefixes admit a zero reward. For the rest of this section, we implicitly assume that this is the case. One way of assuring that this condition is satisfied is to require that the reward function attains positive values only.* \square

Example 2.21 illustrates that we need to resort to lower and upper payoff functions to properly define an average-price-per-reward game. This is not a problem, as strategy and game values for a state of Γ are defined using the framework of zero-sum games in strategic form.

Example 2.21. *Let us consider a very simple game graph consisting of one state belonging to player Max and two edges. One of these edges bears a price of 0, and the other one the price of 1; both edges bear a reward of 1.*

If we use the average-price-per-reward payoff functions, as introduced in this section, and consider an infinite run ω of the form (we only highlight the prices of

the subsequent moves):

$$\underbrace{10}_{2 \cdot 2^0} \underbrace{1100}_{2 \cdot 2^1} \underbrace{11110000}_{2 \cdot 2^2} \dots$$

one can see, after a brief calculation, that $\mathcal{A}_*(\omega) = 1/2$ which is not equal to $\mathcal{A}^*(\omega) = 2/3$. Namely, if we consider subsequences of ω that end on the last 0, just before a 1, their price-per-reward ratio is equal to $\frac{1}{2}$ — this gives us the value of $\mathcal{A}_*(\omega)$. Conversely, we obtain the value of $\mathcal{A}^*(\omega)$ by considering subsequences of ω that end on the last 1, just before a 0, whose price-per-reward ratio converges to $\frac{2}{3}$. In particular, this proves that a single payoff function is not sufficient. \square

The following example should help establish an intuition behind average-price-per-reward games.

Example 2.22. Figure 2.1 illustrates a game graph that models a simple production process. At the start of each production cycle, the manager faces two choices. He can either proceed to produce immediately, or choose to first store electricity, in case of an emergency.

Production costs 8, and storing incurs an additional cost of 4. After the production process has ended, the factory proceeds to sell the produced goods, which results in a reward. Unfortunately, the heavy use of the electricity generator, during the production phase, makes it prone to failures. If no failure occurs then the reward is 32, and it is 24 if the manager decided to store electricity. If a failure does occur, then the rewards are 24 in both cases. Storing electricity, or a failure when no electricity was stored, result in the goods being delivered late, hence the lower reward. Repairs costs 8, and if there is no stored electricity then it has to be bought at a premium of 8.

The states in which the manager makes a choice are assigned to the Min player (denoted using circles). Remaining states, which are not controllable by the manager, model the adversarial environment and are assigned to player Max (denoted by squares).

	OK	Failure
Store	$\frac{1}{2}$	$\frac{5}{6}$
Produce	$\frac{1}{4}$	1

Table 2.1: Price-per-reward ratios for a single production cycle of Figure 2.1.

Consider a single production cycle starting in the “Start” state. Lets assume that the manager decides to proceed to the “Store” state, and that a failure occurs, modelled by the environment choosing to go to the “Repair” state. After the five steps, when the process has finished, the price-per-reward ratio is:

$$\frac{4 + 8 + 8}{24} = \frac{20}{24} = \frac{5}{6}.$$

Table 2.1 summarises all achievable price-per-reward ratios of one production cycle. One can see that, although choosing to produce immediately may result in the lowest ratio, choosing to store first is better in the worst case.

In fact, if from every state the players play indefinitely and optimally, the price-per-reward average will converge to $5/6$. This means that $5/6$ is the value of the average-price-per-reward game in every state. \square

Optimality equations for finite average-price-per-reward games. We now introduce optimality equations that characterise the values of average-price-per-reward games on finite game graphs, and discuss methods for computing their solutions. This formulation differs from that seen earlier in this section, where we used a simpler form to assure a clearer exposition of the concept.

Consider two functions $G, B : \mathcal{S} \rightarrow \mathbb{R}$. We will refer to these functions as *gain* and *bias* respectively. We will formulate a special set of equations, called average-price-per-reward optimality equations, which will have the property that if gain and bias functions satisfy those equations, then $G(s) = \text{Val}(s)$ for all $s \in \mathcal{S}$.

As explained earlier, optimality equations can be seen as a characterisation of

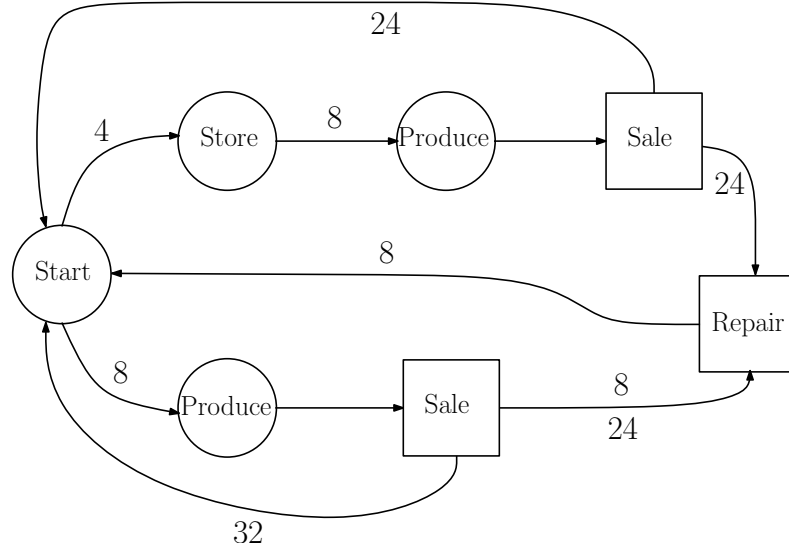


Figure 2.1: A simplified game graph that models a production process in a factory. Each edge has a price (written above the edge) and a reward (written below the edge); we have omitted those that are equal to zero. Circular vertices belong to player Min; the remaining vertices belong to player Max.

game values, and their solutions, if they exist, as witnesses to existence of positional optimal strategies for both players. Variants of these equations were used in the setting of discounted Markov decision processes [Put94], concurrent average-price games [FV97], and hybrid average-price-per-reward games [RLJ10].

Let $G, B : \mathcal{S} \rightarrow \mathbb{R}$. We say that a pair of functions (G, B) is a solution of *average-price-per-reward optimality equations* for Γ , denoted by $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$, if the following conditions hold for all states $s \in \mathcal{S}^{\text{Min}}$:

$$G(s) = \min\{G(s') : (s, s') \in E\}, \quad (2.1)$$

$$B(s) = \min\{\pi((s, s')) - \rho((s, s')) \cdot G(s) + B(s') : (s, s') \in E \text{ and } G(s) = G(s')\}, \quad (2.2)$$

and for all states $s \in \mathbb{S}^{\text{Max}}$:

$$G(s) = \max\{G(s') : (s, s') \in \mathbb{E}\}, \quad (2.3)$$

$$B(s) = \max\{\pi((s, s')) - \rho((s, s')) \cdot G(s) + B(s') : (s, s') \in \mathbb{E} \text{ and } G(s) = G(s')\}. \quad (2.4)$$

The following example aims to establish an intuition behind the optimality equations.

Example 2.23. Recall Example 2.22 and the production model it introduced. Figure 2.2 depicts the same model together with the solution to the optimality equations. Each state has been annotated with its bias, and the gain for every state is $\frac{5}{6}$.

Consider restricting the model to the following cycle “Start”, “Store”, “Produce”, “Sale”, “Repair”, and “Start” again. The average price-per-reward of this cycle is $\frac{5}{6}$. If we formulate the optimality equations for this simple model, then the gain of every state has to be equal — this is because the states are on a cycle. If we denote its value by g , then bias equations have the following form:

$$\begin{aligned} B(\text{Start}) &= 4 - 0 \cdot g + B(\text{Store}) \\ B(\text{Store}) &= 8 - 0 \cdot g + B(\text{Produce}) \\ B(\text{Produce}) &= 0 - 0 \cdot g + B(\text{Sale}) \\ B(\text{Sale}) &= 0 - 24 \cdot g + B(\text{Repair}) \\ B(\text{Repair}) &= 8 - 0 \cdot g + B(\text{Start}) \end{aligned}$$

After summing up the five equalities, we obtain the following:

$$\frac{4 + 8 + 8}{24} = \frac{5}{6} = g.$$

Indeed the gain of each state coincides with the value of the average-price-per-reward

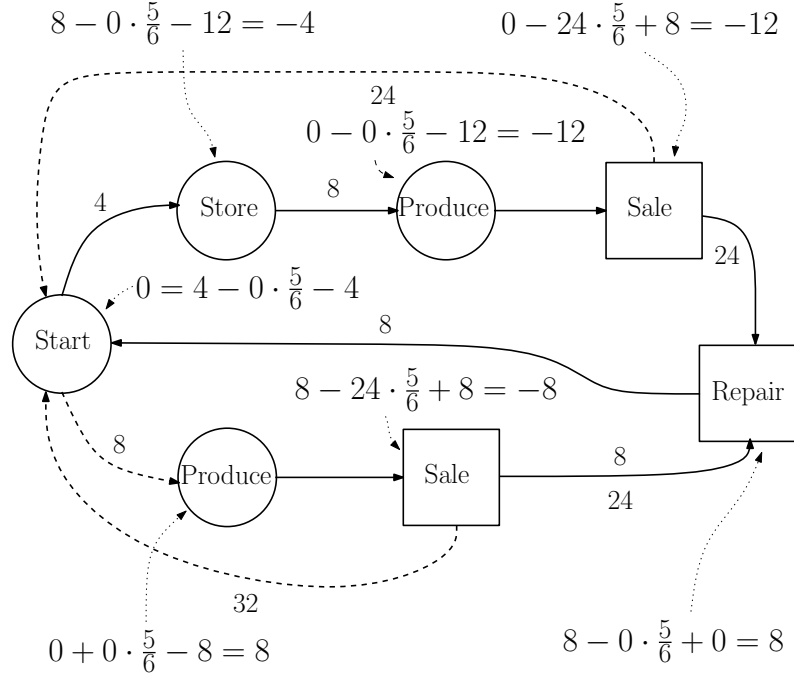


Figure 2.2: Production model from Figure 2.1. Each state is annotated with the value of its bias, under the assumption that the gain of all vertices is $\frac{5}{6}$. Bias of a state is computed based on the bias of the successor state. When choice of successors was available, the solid line highlights the choice made.

game from that state.

Note that the solution to the optimality equations is not necessarily unique, however, the gain of each state is uniquely determined. The one depicted in Figure 2.2 was obtained by arbitrarily setting the bias of the “Start” state to 0. \square

The following theorem formally establishes the properties of optimality equations for average-price-per reward games.

Theorem 2.24. *If $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$ then for every state s , the finite average-price-per-reward game Γ from s is determined and we have $\text{Val}(s) = G(s)$. Moreover, positional optimal strategies exist for both players.*

A more general formulation of this theorem, together with the proof, can be found in Section 4.1.2.

Average-price games are a special case of average-price-per-reward games, where a reward of each edge is equal to 1 for every edge. In average-price games, we are optimising the average price of a move.

The problem of computing the game values for finite average-price games has been extensively studied in the literature [FV97; ZP96; BV07], and strategy improvement algorithms have been proposed [Put94; FV97; BV07]. The complexity lies in the intersection of NP and coNP; pseudo-polynomial [FV97] and strongly sub-exponential [BV07] algorithms exist. These algorithms can be used to compute the solutions to the average-price optimality equations, and can be adapted to compute the solutions to average-price-per-reward optimality equations. A detailed algorithm description can be found in Section 4.2.

Reachability-price games

In this section, we introduce reachability-price games on weighted labelled transition systems — in this setting we disregard the reward information. We will focus on a variant of those games, referred to as finite reachability-price games, which are played on finite game graphs. We will provide a characterisation of game values using a set of equations, referred to as reachability-price optimality equations. The aim of this section is to provide the intuition for reachability-price games, and to explain why reachability-price optimality equations indeed characterise game values. This intuition will be helpful when presenting the results in Chapter 5.

A reachability-price game is played on a game graph Γ (be it a finite game graph, or a game graph defined on a weighted labelled transition system). In this section, we once again abuse the notation and simply write Γ , without specifying the payoff functions, to denote a reachability-price game. As explained in Remark 2.13, in the context of reachability-price games we disregard the reward information of Γ .

In a reachability-price game, player Min wants to assure that a state in a certain designated set of goal states is reached, whereas player Max wants to prevent

it from happening. If player Min can guarantee that a goal state is reached, then he wants to minimise the total cost of doing so; player Max, on the other hand, wants to maximise it. In contrast to average-price-per-reward games, reachability-price games are used in the context of optimising a single execution of a system.

A *reachability-price game* (Γ, F) consists of a finite game graph Γ and a set of *final* states $F \subseteq S$. For a run $\omega \in \text{Runs}$, we define:

$$\text{Stop}(\omega) = \inf_n \{s_n : s_n \in F\},$$

i.e., the index of the first state along the run ω that belongs to one of the goal states. This allows us to complete the definition of a game, by introducing the reachability-price payoff function that is defined as follows:

$$\mathcal{P}(\omega) = \begin{cases} \text{Price}(\omega_{\text{Stop}(\omega)}) & \text{if } \text{Stop}(\omega) < \infty, \\ \infty & \text{otherwise.} \end{cases}$$

The following example should establish an intuition behind the reachability-price games.

Example 2.25. *Figure 2.3 represents a simple model of a production process. In contrast to Example 2.22, we are focusing on modelling a single execution rather than recurring behaviour.*

As in Example 2.22, the production process starts in the “Start” state, and the manager faces two choices. He can either proceed to produce immediately, or choose to first store electricity, in case of an emergency. When the production has finished, the manager may be allowed to sell immediately, or the electricity generator can fail. If no electricity has been stored, the manager has no emergency power to pack the products, and can not sell in time — the production process ends in total failure. If, on the other hand, the manager decided to store electricity, he manages to pack the products and deliver them on time. He does, however, incur an extra

	OK	Failure	Invent
Store	-12	-4	$-\infty$
Produce	-24	$+\infty$	—

Table 2.2: Costs of reaching the “Sale” state from the “Start” state in the game graph of Figure 2.3.

cost related to repairing the generator. There is a third possible outcome of the production stage. If the manager decided to store electricity and no failure occurred, the stored electricity can miraculously shortcircuit the production system, so that it “invents” and assembles a “golden-egg-laying hen”. If that happens, the manager can choose to capitalise on this “invention”, for as long as he chooses too, before deciding to sell. The ultimate goal of the manager is to sell the produced goods, and if this is possible, to minimise the price of doing so. The “prices” of individual steps are shown in Figure 2.3. Positive prices correspond to expenditure, whereas the negative ones to income.

Consider the possible production scenarios starting in the “Start” state. Table 2.25 summarises the cost of reaching “Sale” state depending on the choices of both players.

Notice that $-\infty$ as a value of a strategy, denotes the fact that player Min can ensure an arbitrarily negative payoff. The actual value is not attainable. If player Min chose to stay forever in the “Invent” state, the resulting play would never visit the designated goal location, and hence the payoff would be $+\infty$. \square

Optimality equations for finite reachability-price games. We introduce optimality equations for finite reachability-price games, which characterise values of reachability-price games on finite game graphs, and discuss methods for computing their solutions. This formulation differs from that seen earlier in this section, where we used a simpler form, to assure a clearer exposition of the concept.

Consider two functions $P : S \rightarrow \mathbb{R}$ and $D : S \rightarrow \mathbb{N}$. We will refer to

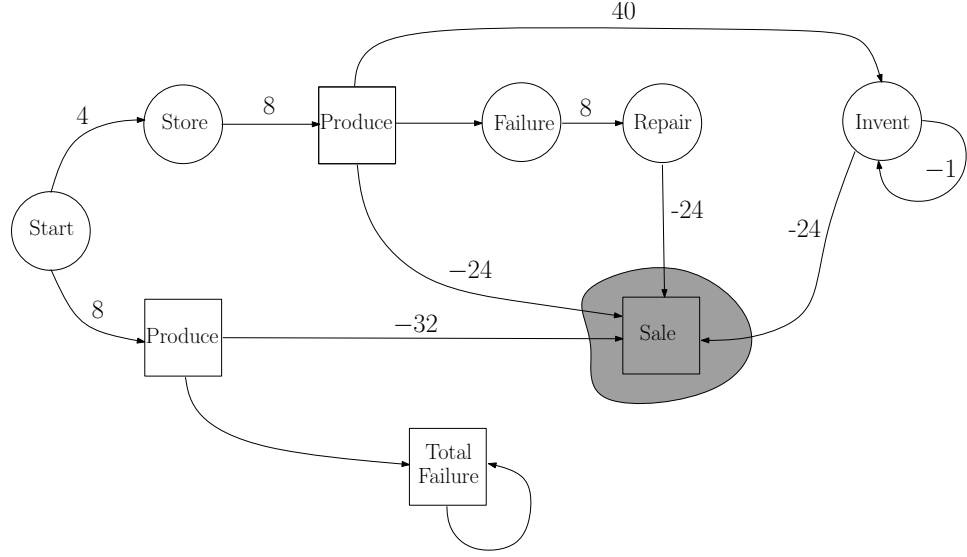


Figure 2.3: A simple game graph that models a single iteration of a production process. The production process starts in the “Start” state, and if all goes well, finishes in the “Sale” state. Edges are annotated with their prices, unless it is equal to zero; the rewards have been omitted. Circle states belong to player Min, and the remaining states belong to player Max. The “Sale” state is the goal state only.

these functions as *price* and *distance*, respectively. We will formulate a special set of equations, called reachability-price optimality equations, which will have the property that if price and distance functions satisfy those equations, then $P(s) = \text{Val}(s)$ for all $s \in S$.

Let $P : S \rightarrow \mathbb{R}$ and $D : S \rightarrow \mathbb{N}$. We say that (P, D) is a solution of the *reachability-price optimality equations* for (Γ, F) , which is denoted by $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma, F)$, if the following conditions hold for all states $s \in S$. If $s \in F$ then $P(s) = D(s) = 0$; if $s \in S^{\text{Min}} \setminus F$ then:

$$\begin{aligned}
 P(s) &= \min \left\{ \pi(s, s') + P(s') \text{ and } (s, s') \in E \right\}, \\
 D(s) &= \min \left\{ 1 + D(s') : P(s) = \pi(s, s') + P(s') \text{ and } (s, s') \in E \right\};
 \end{aligned}$$

and if $s \in S^{\text{Max}} \setminus F$ then:

$$\begin{aligned} P(s) &= \max \left\{ \pi(s, s') + P(s') : (s, s') \in E \right\}, \\ D(s) &= \max \left\{ 1 + D(s') : P(s) = \pi(s, s') + P(s') \text{ and } (s, s') \in E \right\}; \end{aligned}$$

Intuitively, in the equations above $P(s)$ and $D(s)$ capture “optimal price to reach a final state” and “optimal number of steps to reach a final state with optimal price” from state $s \in S$, respectively. The distance equation, which is not strictly necessary for capturing the “optimal price to reach a final state”, will be used to construct almost optimal strategies for both players.

As shown in Example 2.25, in a reachability-price game some states do not admit finite values. The value of negative infinity means that player Min can guarantee an arbitrarily small payoff, whereas the value of positive infinity means that player Max can prevent player Min from reaching a goal state. The reachability-price optimality equations, however, are applicable to the states that admit finite values only. To address this problem, we explain how to determine which states do not admit finite values, and how to restrict the game (Γ, F) to the states that do.

Firstly, we need to determine the set of non-goal states from which player Max can prevent reaching goal. We will denote this set by $W^{\text{Max}} \subseteq S$. In order to compute W^{Max} , one has to compute the winning set of player Max in a reachability-game, which can be easily done in time $O(|S| + |E|)$ for a finite game graph Γ .

Secondly, we need to determine the set of non-goal states from which player Min can guarantee reaching a goal state, and at the same time can guarantee a negative value in an average-price game. We will denote this set by $W^{\text{Min}} \subseteq S \setminus W^{\text{Max}}$. We can compute W^{Min} by first removing all states in W^{Max} from Γ , and then computing the values of an average-price-per-reward game on modified Γ .

It is easy to argue that for all $s \in W^{\text{Max}}$, we have $\text{Val}(s) = +\infty$, and for all $s \in W^{\text{Min}}$ we have $\text{Val}(s) = -\infty$. In the first case, by definition of W^{Max} , player

Max can guarantee that no run from s ever visits a goal state. In the second case, the definition of W^{Min} implies that a family of cycles with a negative total cost is reachable from s . Moreover, player Min can ensure that a cycle in this family is reached and traversed, and that a goal state is reachable from every state on that cycle. Given an arbitrary negative number, in order to assure that the payoff of a run does not exceed it, player Min needs to enforce sufficiently many visits to the aforementioned cycles, and then proceed to the goal state. Notice that, in this case positional strategies are not sufficient. Namely, the strategy needs history to know that the cycle was visited sufficiently many times, and that now it should proceed to the goal state.

The following example should establish the intuition behind the sets W^{Max} and W^{Min} , and the reachability-price optimality equations.

Example 2.26. *Figure 2.4 depicts the game graph that was seen in Example 2.25.*

The “Produce” and “Total Failure” states in the lower part of the game graph have been marked as belonging to the W^{Max} set. Indeed, from those states player Max has a strategy that results in a play that never visits the designated goal location.

The “Invent” state is in the W^{Min} set. If player Min chooses a strategy that always re-enters the “Invent” state, she will be able to achieve a negative average-price of an infinite play that starts in this state. Notice that being able to achieve a negative average-price and to guarantee reaching the designated goal state coincides with the ability of player Min to reach the designated goal state at an arbitrarily small cost.

The “Start” state belongs to $S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$. We will see that the optimality equations indeed characterise the game values. Figure 2.4 gives the values of the price and distance functions, P and D respectively, which are a solution to the

reachability-price optimality equations. The following equalities hold for P :

$$\begin{aligned}
P(\text{"Start"}) &= 4 + P(\text{"Store"}), \\
P(\text{"Store"}) &= 8 + P(\text{"Produce"}), \\
P(\text{"Produce"}) &= P(\text{"Failure"}), \\
P(\text{"Failure"}) &= P(\text{"Repair"}), \\
P(\text{"Repair"}) &= -24 + P(\text{"Sale"}), \\
P(\text{"Sale"}) &= 0
\end{aligned}$$

and the following hold for D :

$$\begin{aligned}
D(\text{"Start"}) &= 1 + D(\text{"Store"}), \\
D(\text{"Store"}) &= 1 + D(\text{"Produce"}), \\
D(\text{"Produce"}) &= 1 + D(\text{"Failure"}), \\
D(\text{"Failure"}) &= 1 + D(\text{"Repair"}), \\
D(\text{"Repair"}) &= 1 + D(\text{"Sale"}). \\
D(\text{"Sale"}) &= 0
\end{aligned}$$

If we add up, and simplify the equalities we will see that $P(\text{"Start"}) = -4$ and $D(\text{"Start"}) = 5$. Moreover, recall that according to Table 2.25 this value is attained when player Min chooses "Store" in the "Start" state and player Max chooses "Failure" in the "Produce" state. Notice that this coincides with the fact that $P(\text{"Start"}) = P(\text{"Store"})$ and $P(\text{"Produce"}) = P(\text{"Failure"})$ (when augmented with +1 on the left hand side, the same equalities hold for D). \square

Let $S^{\text{fin}} = S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$ and let Γ^{fin} be the weighted game graph obtained from Γ by restricting to the set of states S^{fin} . The following theorem establishes that reachability-price optimality equations characterise the values of a

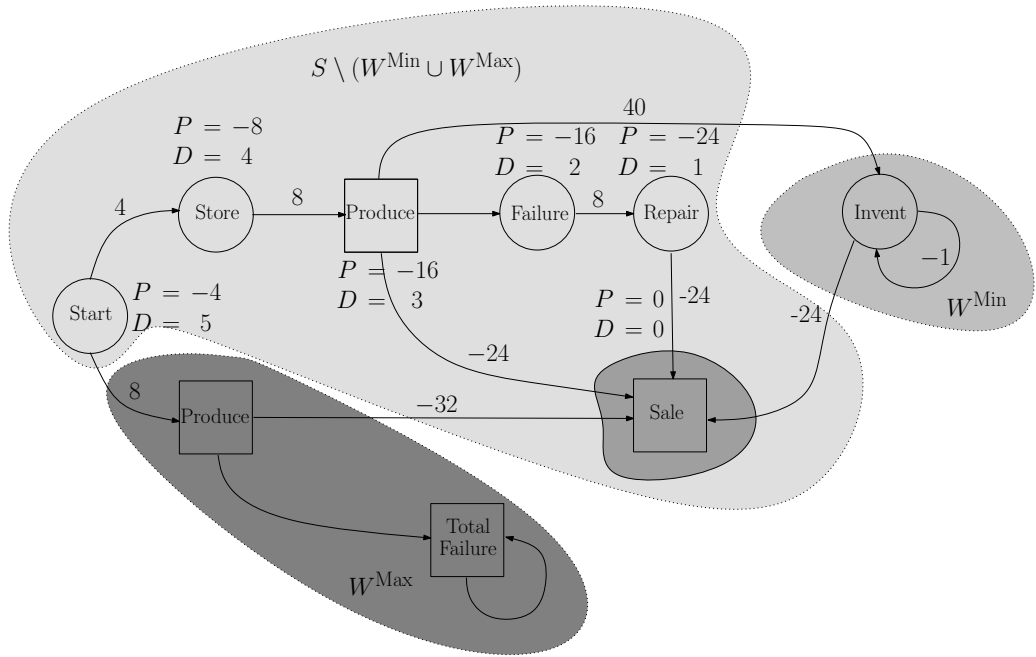


Figure 2.4: The game graph of Figure 2.3. The states are partitioned into the sets W^{Max} , W^{Min} , and $S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$. The states in the $S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$ are annotated with the values of the P and D functions, which satisfy the reachability-price optimality equations.

reachability-price game $(\Gamma^{\text{fin}}, \mathbf{F})$.

Theorem 2.27. *For every finite game graph Γ , there is a pair of functions $P : \mathcal{S}^{\text{fin}} \rightarrow \mathbb{R}$ and $D : \mathcal{S}^{\text{fin}} \rightarrow \mathbb{N}$, such that $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}}, \mathbf{F})$, and for every state $s \in \mathcal{S}^{\text{fin}}$, the reachability-price game Γ from s is determined and $\text{Val}(s) = P(s)$.*

A more general formulation of this theorem, together with the proof, can be found in Section 5.1.2. Strategy improvement algorithms [Put94; FV97; JT07] can be used to compute the solutions to the reachability-price optimality equations.

2.3 Hybrid automata

The purpose of this section is to introduce hybrid automata and their semantics. This background will be used to informally define the hybrid automata related concepts that are of interest to this thesis: hybrid-automata with strong resets, timed automata, and hybrid games.

Hybrid automata, an extension of finite automata, are a formalism for modelling hybrid systems. The discrete component of the system is modelled by a finite control graph, whereas the continuous one is modelled by a set of continuous variables. The changes to the continuous variable values are governed by a flow function, which in turn depends on the state of the control graph. The state of the hybrid automaton comprises of the state of the underlying control graph and the values of the continuous variables.

Firstly, we introduce the general definition of a hybrid automaton and its semantics, and show a simple example. Secondly, we proceed to introduce the two subclasses of hybrid automata: hybrid automata with strong resets (Section 2.3.1) and timed automata (Section 2.3.2) that will be considered in this thesis. The former class admits complex continuous dynamics at the cost of strong assumptions on the discrete changes, and the latter captures only simple continuous behaviours but does not place such strong restriction on the discrete changes. We conclude this

section with an introduction to games on hybrid automata (Section 2.3.3).

We will now present the classical definition of a hybrid automaton and its semantics, as found in the literature [Hen96]. Our definition differs slightly from the one broadly used, as we do not distinguish initial states, and we do not assign labels to control actions. These elements have been omitted since they are not important for our considerations.

Definition 2.28 (Hybrid Automaton). *A hybrid automaton \mathcal{H} consists of:*

Continuous variables *A finite set $X = \{x_1, \dots, x_n\}$ of real-valued variables. The number n is called the dimension of \mathcal{H} . The set $\dot{X} = \{\dot{x}_1, \dots, \dot{x}_n\}$ denotes the set of first derivatives of elements of X during the continuous change. Finally, we write X' for the set $\{x'_1, \dots, x'_n\}$ of primed variables, which represent values at the conclusion of a discrete change.*

Control graph *A finite directed graph (L, A) . The locations in L are called control modes. The actions in A are called control actions.*

Invariant and flow conditions *Two vertex-labelling functions inv and flow that assign to each control mode $l \in L$ two predicates. Each invariant condition $\text{inv}(l)$ is a predicate whose free variables are from X . Each flow condition $\text{flow}(l)$ is a predicate whose free variables are from $X \cup \dot{X}$.*

Jump conditions *An action-labelling function jump that assigns to each control switch $a \in A$ a predicate. Each jump condition $\text{jump}(a)$ is a predicate whose free variables are from $X \cup X'$.*

The following example illustrates the definition of a hybrid automaton.

Example 2.29. *Recall the example of a boiler with a thermostat that was introduced in the beginning of Section 1.1. The physical properties of the boiler are as follows: the water never freezes, and the heater can not heat the water above 86.4°C . For the*

sake of simplicity, we approximate the heating and cooling behaviour of the boiler, as shown in Figure 1.2, using quadratic functions.

The function $h : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ captures how the temperature changes over time:

$$h(t) = \begin{cases} -\frac{6 \cdot t^2}{1000} + \frac{36 \cdot t}{25} & \text{if } t \leq 120, \\ 86.4 & \text{if } t > 120. \end{cases}$$

If the boiler is in the cooling mode and the temperature of the water is 86.4°C then it takes just over 48 hours, i.e., 2800 minutes, for it to cool down to 0°C . The function $c : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ captures how the temperature changes over time:

$$c(t) = \begin{cases} -\frac{27 \cdot t^2}{245000} + \frac{54 \cdot t}{175} + 86.4 & \text{if } t \leq 2800, \\ 0 & \text{if } t > 2800. \end{cases}$$

We will now show how to model this simple hybrid system using hybrid automata. We will use a single automaton that has two control locations that model the two modes of operation, and one continuous variable that models the temperature. The control actions will correspond to changes in the mode of operation. The automaton is defined as follows:

- $\mathbf{X} = \{x\}$ is the finite set of real-valued variables. It contains a single variable (temperature).
- $\langle \mathbf{L}, \mathbf{A} \rangle$ is the control graph. There are two control locations $\{\text{ON}, \text{OFF}\}$, and two control actions $\{(\text{ON}, \text{OFF}), (\text{OFF}, \text{ON})\}$. The control locations correspond to the modes of operation, i.e., heating and cooling, respectively. The control actions correspond to the changes of operation mode, i.e., switching off and switching on, respectively.

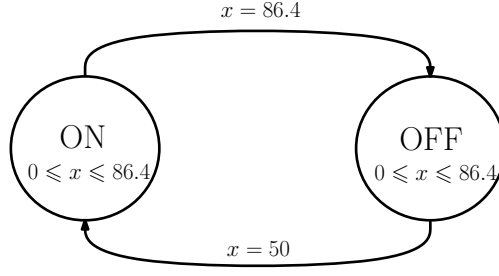


Figure 2.5: Hybrid automaton that models the boiler.

- *The invariant condition is given as:*

$$\text{inv}(\text{ON}) = \text{inv}(\text{OFF}) = 0 \leq x \leq 86.4.$$

It ensures that the physical properties of the boiler are taken into account.

- *The flow conditions are defined as follows:*

$$\begin{aligned} \text{flow}(\text{ON})(x, \dot{x}) &= (\exists t. x = h(t) \wedge \dot{x} = h'(t)), \\ \text{flow}(\text{OFF})(x, \dot{x}) &= (\exists t. x = c(t) \wedge \dot{x} = c'(t)), \end{aligned}$$

where the functions h and c are defined as in the beginning of this example.

The flow conditions capture the changes to the water temperature.

- *The jump condition is given as:*

$$\text{jump}((\text{ON}, \text{OFF})) = (x = 86.4 \wedge x = x'),$$

and

$$\text{jump}((\text{OFF}, \text{ON})) = (x = 50 \wedge x = x').$$

The jump conditions capture the changes to the boiler's mode of operation.

Figure 2.5 depicts the hybrid automaton we have just defined.

□

The semantics of a hybrid automaton will be given in terms of a doubly-weighted labelled transition system (see Definition 2.12). The state of the transition system will consist of the current control location and the current valuation of the real-valued variables. There will be two types of transitions: continuous and discrete. A continuous transition results in a change of the valuation of the real-valued variables, and a discrete transition results in a change of the current control location. The admissible continuous transitions are specified by the flow and inv conditions, and the admissible discrete transitions are specified by the jump and inv predicates. An execution of a hybrid automaton \mathcal{H} is modelled by a run of its transition system $\mathcal{T}_{\mathcal{H}}$. When talking about behaviours of a hybrid automaton, we will, in fact, be referring to the behaviours of its transition system. In particular, we will often refer to a state of a hybrid automaton, or to a run of a hybrid automaton, but in fact we will be referring to its transition system.

Remark 2.30. *We will write $\mathsf{T} = \mathbb{R}_{\geq 0}$ to denote the set of all time durations, when referring to the duration of a continuous transition.* \square

Definition 2.31 (Transition semantics of hybrid automata). *The transition semantics of a hybrid automaton \mathcal{H} is given by a labelled transition system $\mathcal{T}_{\mathcal{H}}$ consisting of:*

- *the set of states $\mathsf{S}_{\mathcal{H}} \subseteq \mathsf{L} \times \mathbb{R}^n$ such that: $(\ell, x) \in \mathsf{S}_{\mathcal{H}}$ iff $\text{inv}(\ell)[\mathsf{X} := x]$ is true.*
- *the set of transition labels $\Lambda_{\mathcal{H}} \subseteq \mathsf{A} \cup \mathsf{T}$, and the transition relation defined as follows:*

Continuous transition *for every $t \in \mathsf{T}$, there is a transition $(\ell, x) \xrightarrow{t} (\ell', x')$ iff $\ell = \ell'$ and there is a differentiable function $f : [0, t] \rightarrow \mathbb{R}^n$, with the first derivative $\dot{f} : (0, t) \rightarrow \mathbb{R}^n$, such that (1) $f(0) = x$ and $f(t) = x'$, and (2) for all $t' \in (0, t)$ both $\text{inv}(\ell)[\mathsf{X} := f(t')]$ and $\text{flow}(\ell)[\mathsf{X}, \dot{\mathsf{X}} := f(t'), \dot{f}(t')]$ are true. The function f will be referred to as a witness to the transition $(\ell, x) \xrightarrow{t} (\ell', x')$.*

Discrete transition for every $a \in \mathbf{A}$, there is a transition $(\ell, x) \xrightarrow{a} (\ell', x')$ iff there is a control action $(\ell, \ell') \in \mathbf{A}$ such that $\text{jump}(a)[\mathbf{X}, \mathbf{X}' := x, x']$ is true.

Remark 2.32. In Definition 2.31 we have omitted the definition of the two weight functions, which are an integral part of a doubly-weighted labelled transition system. The definition of these function will vary, depending on the subclass of hybrid automata considered, and the purpose of Definition 2.31 was to give a preliminary understanding. In Chapters 3 and 5, the definitions of those functions will be provided. \square

Remark 2.33. We only allow runs that admit only finitely many consecutive timed transitions. Note that this requirement does not exclude Zeno runs, i.e., runs that admit infinitely many discrete transitions, but are of finite duration. \square

The following example will show how, given a hybrid automaton, to construct its transition system.

Example 2.34. Given the hybrid automaton from Example 2.29, which models a boiler, we will construct its transition system. Lets assume that: the reward of all transitions is zero; the price of discrete transitions is zero; the cost of heating is 10 cents per minute, i.e., linear in time; the cost of cooling is zero. If functions $h, c : \mathbb{T} \rightarrow \mathbb{R}$ are defined as in Example 2.29, then the transition system is defined as follows.

- The state space is equal to $\mathbf{S} = \{\text{ON}, \text{OFF}\} \times [0, 86.4]$.
- The set of transition labels is equal to $\Lambda = \{(\text{ON}, \text{OFF}), (\text{OFF}, \text{ON})\} \cup \mathbb{T}$. The transition relation is defined as follows.

Continuous transition. Given $t \in \mathbb{T}$, a continuous transition $(\text{ON}, x_1) \xrightarrow{t} (\text{OFF}, x_2)$ is feasible if $(\text{ON}, x_1), (\text{ON}, x_2) \in \mathbf{S}$, and there exists $t_1 \in \mathbb{T}$ such that $h(t_1) = x_1$ and $h(t_1 + t) = x_2$. If this is the case, then the

function $f : [0, t] \rightarrow \mathbb{R}$, the requested witness to the transition, is given as $f(t') = h(t_1 + t')$; f is differentiable over $(0, t)$ because h is differentiable over its whole domain.

Given $t \in \mathbb{T}$, a continuous transition $(\text{OFF}, x_1) \xrightarrow{t} (\text{OFF}, x_2)$ is feasible if $(\text{OFF}, x_1), (\text{OFF}, x_2) \in \mathbb{S}$, and there exists $t_1 \in \mathbb{T}$ such that $c(t_1) = x_1$ and $c(t_1 + t) = x_2$. If this is the case, then the function $f : [0, t] \rightarrow \mathbb{R}$, the requested witness to the transition, is given as $f(t') = c(t_1 + t')$; f is differentiable over $(0, t)$ because c is differentiable over its whole domain.

Discrete transition. There are only two feasible discrete transitions. These are:

$$(\text{ON}, 86.4) \xrightarrow{(\text{ON}, \text{OFF})} (\text{OFF}, 86.4) \text{ and } (\text{OFF}, 50) \xrightarrow{(\text{OFF}, \text{ON})} (\text{ON}, 50).$$

- For every transition $s \xrightarrow{\lambda} s'$ the price function is defined as

$$\pi(s, \lambda, s') = \begin{cases} 10 \cdot \lambda & \text{if } s = (\text{ON}, x) \text{ and } \lambda \in \mathbb{T} \\ 0 & \text{otherwise.} \end{cases}$$

and the reward function is defined as $\rho(s, \lambda, s') = 0$.

Consider the state $s = (\text{ON}, 0)$, a possible run from that state could look like this:

$$\begin{aligned} (\text{ON}, 0) &\xrightarrow{30} (\text{ON}, 37.8) \xrightarrow{90} (\text{ON}, 86.4) \xrightarrow{(\text{ON}, \text{OFF})} (\text{OFF}, 86.4) \xrightarrow{670} \\ &(\text{OFF}, 50) \xrightarrow{(\text{OFF}, \text{ON})} (\text{ON}, 50) \xrightarrow{78} (\text{ON}, 86.4) \dots \end{aligned}$$

In case of the first continuous transition, the requested t_1 is equal to 0, and the witness is exactly the h function introduced in Example 2.29. In case of the second continuous transition, the requested t_1 is equal to 30, and the witness is $f(t') = h(t' + 30)$. For example, the prices of the first and second transitions are $10 \cdot 30 = 300$

and $90 \cdot 10 = 900$, respectively, but the prices of the third and fourth transitions are equal 0. \square

2.3.1 Hybrid automata with strong resets

Hybrid automata with strong resets are a subclass of hybrid automata [LPS00; BM05; BBC09]. The name comes from the strong assumption on the definition of the jump predicate. Although complex flow predicates are allowed, a discrete transition results in the values of all continuous variables being non-deterministically “reset”.

Suppose that the hybrid automaton \mathcal{H} has the property that, for all actions a , the values of $x, x' \in \mathbb{R}^n$ satisfying the predicate $\text{jump}(a)$ are mutually independent, and depend only on a . More formally, for every edge a , there exist two sets $G(a) \subseteq \mathbb{R}^n$ and $R(a) \subseteq \mathbb{R}^n$ such that, $\text{jump}(a)[X, X' := x, x']$ is true iff $x \in G(a)$ and $x' \in R(a)$. The two sets are referred to as the *guard* and *reset* sets of an edge, respectively, the property is referred to as the *strong resets* property, and a hybrid automaton satisfying it is referred to as a *hybrid automaton with strong resets*.

We now define hybrid automata with strong resets. In this section we present the classical definition so that the reader can clearly see how the class of hybrid automata with strong resets relates to other classes of hybrid automata. In Chapter 5 we will provide a less typical definition. The definition presented there has been tailored to allow for cleaner exposition of our results. This is achieved through “hiding” the flow condition of the automaton in the definition of the guard predicates.

Definition 2.35 (Hybrid automaton with strong resets). *An automaton with strong resets \mathcal{H} consists of the following components:*

- X is the finite set of real-valued variables,
- (L, A) is the finite control graph,

- flow and inv are two-location labelling functions that assign to each control location a predicate whose free variables are from $\mathbf{X} \cup \dot{\mathbf{X}}$ and \mathbf{X} , respectively.
- \mathbf{G} and \mathbf{R} are two action-labelling functions that assign to each control action two predicates whose free variables are in from \mathbf{X} and \mathbf{X}' respectively.

The semantics for a hybrid automaton with strong resets is given in terms of a doubly-weighted transition system. The transition system differs from the transition system of the general hybrid automaton in the way discrete transitions are defined. As we have done before, we omit the specification of the price and reward functions. The details of how these functions are defined can be found in Chapter 5.

Definition 2.36 (Transition semantics of hybrid automata with strong resets). *The transition semantics of a hybrid automaton with strong resets \mathcal{H} is given by a labelled transition system $\mathcal{T}_{\mathcal{H}}$, consisting of:*

- the set of states $\mathbf{S}_{\mathcal{H}} \subseteq \mathbf{L} \times \mathbb{R}^n$ such that: $(\ell, x) \in \mathbf{S}$ iff $\text{inv}(\ell)[\mathbf{X} := x]$ is true.
- the set of transition labels $\Lambda_{\mathcal{H}} \subseteq \mathbf{A} \cup \mathbf{T}$, and the transition relation defined as follows:

Continuous transition for every $t \in \mathbf{T}$, there is a transition $(\ell, x) \xrightarrow{t} (\ell', x')$ iff $\ell = \ell'$ and there is a differentiable function $f : [0, t] \rightarrow \mathbb{R}^n$ with the first derivative $\dot{f} : (0, t) \rightarrow \mathbb{R}^n$, such that (1) $f(0) = x$ and $f(t) = x'$, and (2) for all $t' \in (0, t)$ both $\text{inv}(\ell)[\mathbf{X} := f(t')]$ and $\text{flow}(\ell)[\mathbf{X}, \dot{\mathbf{X}} := f(t'), \dot{f}(t')]$ are true. The function f will be referred to as a witness to the transition $(\ell, x) \xrightarrow{t} (\ell', x')$.

Discrete transition for every $a \in \mathbf{A}$, there is a transition $(\ell, x) \xrightarrow{a} (\ell', x')$ iff there is a control action $(\ell, \ell') \in \mathbf{A}$ such that $\mathbf{G}(a)[\mathbf{X} := x]$ and $\mathbf{R}(a)[\mathbf{X}' := x']$ are true.

The following example illustrates the definition of hybrid automata with strong resets.

Example 2.37. *We will now show how to model the boiler, introduced in Example 2.29, using hybrid automata with strong resets. The key difference, with respect to the automaton defined in Example 2.29, will be in the definition of the jump predicate. Instead of the jump predicate we will be using the aforementioned G and R sets. The sets are defined as follows:*

- *For the control action (ON, OFF) the guard and reset sets are equal to:*

$$G((\text{ON}, \text{OFF})) = \{86.4\} \text{ and } R((\text{ON}, \text{OFF})) = \{86.4\},$$

respectively.

- *For the control action (OFF, ON) the guard and reset sets are equal to:*

$$G((\text{OFF}, \text{ON})) = \{50\} \text{ and } R((\text{OFF}, \text{ON})) = \{50\},$$

respectively. The guard and reset sets specify the jump condition of the classical hybrid automaton. It would have been defined as $\text{jump}(a)[X, X' := x, x']$ iff $x \in G(a)$ and $x' \in R(a)$.

Figure 2.6 depicts the hybrid automaton we have just defined.

The transition system induced by the hybrid automaton with strong resets we have defined is the same as the one induced by the hybrid automaton from Example 2.29, i.e., the transition system defined in Example 2.34. \square

2.3.2 Timed automata

Rather than placing strong restrictions on discrete transitions, we can restrict the continuous behaviour of the automaton. Timed automata, first introduced by Alur and Dill [AD94], are a subclass of hybrid automata that admit continuous transitions occurring at a uniform and fixed constant rate only.

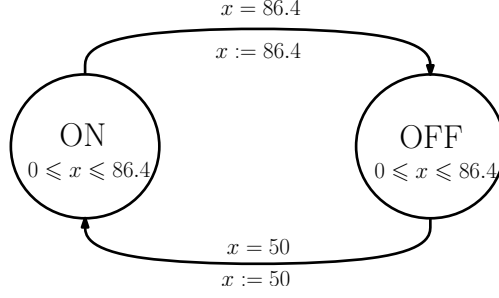


Figure 2.6: A graphical representation of the hybrid automaton with strong resets that models the boiler, as introduced in Example 2.37.

In a timed automaton, the continuous variables admit non-negative values only, and are referred to as clocks. Their values change uniformly at a fixed and constant rate of 1, i.e., for all locations $\ell \in \mathbf{L}$ and $x \in \mathbb{R}_{\geq 0}$, the closed predicate $\text{flow}(\ell)[\mathbf{X}, \dot{\mathbf{X}} := x, \dot{x}]$ is true iff $\dot{x}_i = 1$ for all $i \in \{1, \dots, n\}$. Given a transition $(\ell, x) \xrightarrow{t} (\ell, x')$, for some $t \in \mathbb{R}_{\geq 0}$, the function $f : [0, t] \rightarrow \mathbb{R}^n$ that is a witness to this continuous transition is defined as $f(t') = (x_1 + t', \dots, x_n + t')$, i.e., it is a linear function. Therefore, the flow condition will be omitted from the definition of a timed automaton, as it is given implicitly.

The distinguishing feature of timed automata is a certain type of continuous dynamics, however they also place restrictions on the discrete transitions. Namely, the sets of clock values, satisfying the $\text{jump}(a)$ predicate, form closed and convex sets, and as such, can be described using conjunctions of inequalities that compare clock values to integers.

Given a set of clocks \mathbf{X} , consider predicates of the form $x_i \bowtie c$ and $x_i - x_j \bowtie c$, where c is a non-negative integer constant, \bowtie is one of the following relation symbols $\{=, \leq, \geq, <, >\}$, and x_i and x_j are clocks, with i, j ranging over $\{1, \dots, |\mathbf{X}|\}$. These predicates are referred to as *clock constraints*. We will write $\mathcal{B}(\mathbf{X})$ to denote the set of all clock constraints over the set of clocks \mathbf{X} .

The jump predicate is not explicitly stated in the classical definitions of timed automata. Instead, an action a is a tuple (ℓ, g, Z, ℓ') , where ℓ and ℓ' are locations,

$Z \subseteq \mathbf{X}$ is referred to as the *reset set*, and g is the *guard* that is specified by a conjunction of clock constraints. The reset set specifies which clocks are to be reset to the value of 0 after the discrete transition corresponding to a occurs, the values of the remaining clocks do not change. We can define a predicate reset, whose free variables are from $\mathbf{X} \cup \mathbf{X}'$, such that the closed predicate $\text{reset}(a)[\mathbf{X}, \mathbf{X}' := x, x']$ is true iff

$$x'_i = \begin{cases} 0 & \text{if } x_i \in Z, \\ x_i & \text{otherwise,} \end{cases}$$

for $i \in \{1, \dots, n\}$, denoted by $\mathbf{X}' = \mathbf{X}[Z := 0]$. In a hybrid automaton the action (ℓ, g, Z, ℓ') would have been represented by an action $a = (\ell, \ell')$ and a predicate $\text{jump}(a) = g(a) \wedge \text{reset}(a)$.

We now present the formal definition of a timed automaton. Unlike the general definition of a hybrid automaton presented at the beginning of this section, our definition of a timed automaton will feature a price information — it will be a weighted timed automaton. This is the model we will be considering in Chapter 3, where we will be considering reachability-price games on single-clock timed automata (recall that in reachability-price games, only price information is being taken into account). We also introduce the notion of an urgent location, i.e., control location in which no continuous transitions can occur. In a general hybrid automaton, one did not have explicitly distinguish such locations as they could be specified using the flow predicate. In timed automata the flow predicate is absent, and hence, to indicate which locations are urgent, we use the notion of an urgency mapping.

Definition 2.38 (Timed automaton). *A weighted timed automaton, or simply a timed automaton \mathcal{A} consists of the following components:*

- \mathbf{X} , the finite set of real-valued variables, referred to as clocks,
- \mathbf{L} , the finite set of control locations,
- $\mathbf{A} \subseteq \mathbf{L} \times \mathcal{B}(\mathbf{X}) \times 2^{\mathbf{X}} \times \mathbf{L}$, the finite set of control actions,

- $\text{inv} : \mathbf{L} \rightarrow \mathcal{B}(\mathbf{X})$, the invariant condition,
- $\text{urg} : \mathbf{L} \rightarrow \{0, 1\}$, the urgency mapping,
- $\pi : \mathbf{L} \cup \mathbf{A} \rightarrow \mathbb{N}$, the price function.

The semantics of a timed automaton is given in terms of a transition system. In this case it will be a weighted transition system, as the timed automaton features only a price function.

Definition 2.39 (Transition semantics of timed automata). *The transition semantics of a weighted timed automaton \mathcal{A} is given by a weighted labelled transitions system $\mathcal{T}_{\mathcal{A}}$, defined as follows:*

- $\mathbf{S}_{\mathcal{A}} \subseteq \mathbf{L} \times \mathbb{R}_{\geq 0}^{|\mathbf{X}|}$ is the state space such that $\text{inv}(\ell)[\mathbf{X} := x]$ is true, for every $(\ell, x) \in \mathbf{S}_{\mathcal{A}}$,
- $\Lambda_{\mathcal{A}} \subseteq \mathbf{A} \cup \mathbf{T}$ is the set of labels. For a label $\lambda \in \Lambda_{\mathcal{A}}$, there is a transition $(\ell, x) \xrightarrow{\lambda} (\ell', x')$ iff one of the following is true:

Continuous transition $\lambda = t \in \mathbf{T}$, $\text{urg}(\ell) = 0$, i.e., the location is non-urgent, for every $t' \in (0, t)$ we have $\text{inv}[\mathbf{X} := x + t']$ is true, $\ell = \ell'$, and $x' = x + t$.

Discrete transition $\lambda = (\ell, g, Z, \ell') \in \mathbf{A}$, $g[\mathbf{X} := x]$ holds, and $x' = x[Z := 0]$.

- $\pi_{\mathcal{A}} : \mathbf{S}_{\mathcal{A}} \times \Lambda_{\mathcal{A}} \times \mathbf{S}_{\mathcal{A}} \rightarrow \mathbb{R}$ is the price function defined as follows:

$$\pi_{\mathcal{A}}((\ell, x) \xrightarrow{\lambda} (\ell', x')) = \begin{cases} \pi(\lambda) & \text{if } \lambda \in \mathbf{A}, \\ \lambda \cdot \pi(\ell) & \text{if } \lambda \in \mathbf{T}. \end{cases}$$

We now give an example that demonstrates how timed automata differ from general hybrid automata.

Example 2.40. Recall the hybrid automaton from Example 2.29, which modelled the boiler. We will now show how to model the boiler using a timed automaton. The continuous variable, i.e., the clock, will be used to keep track of time in the boiler's cycle of operation. There will be a direct correspondence between the clock value and the temperature of the water in the boiler. We have to "encode" the temperature in this way because timed automata do not provide the means to directly model the complex continuous dynamics.

To construct the timed automaton, we will use three rather than two control locations. The first location will correspond to the boiler being in the heating mode, and the temperature being at most 50°C . The boiler is in this state only in the very beginning, and after the maximum temperature is reached, the temperature never falls below that threshold. The second control location will correspond to the heating mode of operation, and the temperature being at least 50°C . The third and last location will correspond to the cooling mode. Once we complete the definition of the automaton, we will explain how to retrieve the state of the boiler from the state of the automaton. The timed automaton is defined as follows.

- $X = \{x\}$ is the finite set of clocks. It contains a single clock.
- $L = \{\text{Start}, \text{ON}, \text{OFF}\}$ is the finite set of control locations. The three control locations correspond to the boiler's modes of operation, i.e., heating (the temperature is up to to 50°C), heating (the temperature is at least 50°C), and cooling, respectively.
- The finite set of control actions is equal to:

$$A = \left\{ (\text{Start}, \{x\}, x = 42, \text{ON}), \right. \\ \left. (\text{ON}, \emptyset, x = 78, \text{OFF}), (\text{OFF}, \{x\}, x = 748, \text{ON}) \right\}$$

The three control actions correspond to the changes of operation mode, i.e.,

switching on and off.

- The invariant condition for the Start location is given as:

$$\text{inv}(\text{Start}) = (x \geq 0) \wedge (x \leq 42);$$

for the ON location the invariant condition is given as:

$$\text{inv}(\text{ON}) = (x \geq 0) \wedge (x \leq 78);$$

for the OFF location the invariant condition is given as:

$$\text{inv}(\text{OFF}) = (x \geq 78) \wedge (x \leq 748).$$

- All control locations are non-urgent, i.e., $\text{urg}(\ell) = 0$ for every $\ell \in \mathbf{L}$.
- The price function is defined as follows:

$$\pi(x) = \begin{cases} 10 & \text{if } x = \text{Start} \text{ or } x = \text{ON}, \\ 0 & \text{if } x = \text{OFF} \text{ or } x \in \mathbf{A}. \end{cases}$$

Figure 2.7 depicts the timed automaton we have just defined. We will now define the transition system induced by this automaton.

- The state space is equal to:

$$\mathbf{S} = (\{\text{Start}\} \times [0, 42]) \cup (\{\text{ON}\} \times [0, 78]) \cup (\{\text{OFF}\} \times [78, 748])$$

- The set of transition labels is equal to $\Lambda = \mathbf{A} \cup \mathbf{T}$.

Continuous transition. There are three kinds of continuous transitions, and the feasible transitions are as follows. Given $t \in \mathbf{T}$, a continuous transi-

tion $(\text{Start}, x) \xrightarrow{t} (\text{Start}, x + t)$ is feasible iff $x, x + t \in [0, 42]$.

Given $t \in \mathbb{T}$, a continuous transition $(\text{ON}, x) \xrightarrow{t} (\text{ON}, x + t)$ is feasible iff $x, x + t \in [0, 78]$.

Given $t \in \mathbb{T}$, a continuous transition $(\text{OFF}, x) \xrightarrow{t} (\text{OFF}, x + t)$ is feasible iff $x, x + t \in [78, 748]$.

Discrete transition. There are three discrete transitions, and they are:

$$\begin{aligned} (\text{Start}, 42) & \xrightarrow{(\text{Start}, \{x\}, x=42, \text{ON})} (\text{ON}, 0) \\ (\text{ON}, 78) & \xrightarrow{(\text{ON}, \emptyset, x=78, \text{OFF})} (\text{OFF}, 78) \\ (\text{OFF}, 748) & \xrightarrow{(\text{OFF}, \{x\}, x=678, \text{ON})} (\text{ON}, 0) \end{aligned}$$

- $\pi : \mathbf{S} \times \Lambda \times \mathbf{S} \rightarrow \mathbb{R}$ is the price function defined as follows:

$$\pi((\ell, x) \xrightarrow{\lambda} (\ell', x')) = \begin{cases} 0 & \text{if } \lambda \in \mathbf{A}, \\ 0 & \text{if } \ell = \text{OFF}, \\ \lambda \cdot 10 & \text{otherwise.} \end{cases}$$

Recall the definitions of the functions h and c , which characterised the heating and cooling properties of the boiler, introduced in Example 2.29. We now explain how the states of the timed automaton, defined in this example, correspond to states of the hybrid automaton defined in Example 2.29. A state (Start, x) corresponds to the state $(\text{ON}, h(x))$ in the hybrid automaton, The state (ON, x) corresponds to the state $(\text{ON}, h(x + 42))$ in the hybrid automaton, and the state (OFF, x) corresponds to a state $(\text{OFF}, c(x - 78))$ in the hybrid automaton.

Consider the state $s = (\text{ON}, 0)$, an example run of the timed automaton could

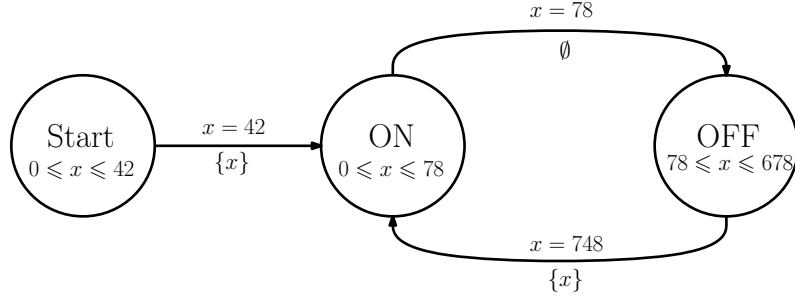


Figure 2.7: Timed automaton that models the boiler.

be as follows:

$$\begin{aligned}
 (\text{Start}, 0) &\xrightarrow{30} (\text{Start}, 30) \xrightarrow{12} (\text{Start}, 42) \xrightarrow{(\text{Start}, \{x\}, x=42, \text{ON})} (\text{ON}, 0) \xrightarrow{78} \\
 (\text{ON}, 78) &\xrightarrow{(\text{ON}, \emptyset, x=78, \text{OFF})} (\text{OFF}, 78) \xrightarrow{670} (\text{OFF}, 748) \xrightarrow{(\text{OFF}, \{x\}, x=748, \text{ON})} \\
 &(\text{ON}, 0) \xrightarrow{78} (\text{ON}, 78) \dots
 \end{aligned}$$

Notice that this run corresponds to the following run, in the hybrid automaton defined in Example 2.34:

$$\begin{aligned}
 (\text{ON}, 0) &\xrightarrow{30} (\text{ON}, 37.8) \xrightarrow{12} (\text{ON}, 50) \xrightarrow{78} \\
 (\text{ON}, 86.4) &\xrightarrow{(\text{ON}, \text{OFF})} (\text{OFF}, 86.4) \xrightarrow{670} (\text{OFF}, 50) \xrightarrow{(\text{OFF}, \text{ON})} \\
 &(\text{ON}, 50) \xrightarrow{78} (\text{ON}, 86.4) \dots
 \end{aligned}$$

The correspondence goes further. The accumulated price of corresponding finite prefixes in both runs are equal to each other. \square

2.3.3 Games on hybrid automata

To define games on hybrid automata we will use the concept of a game on a doubly-weighted labelled transition system, as introduced in Section 2.2.2. We will present two approaches to defining those games: turn based games, and quasi-concurrent

games. In the first approach, we partition the set of control locations between players, and this partition naturally induces a game graph on the transition system. In the second approach, it is the set of control actions which is partitioned between players; there is no need to define a game graph structure. In this setting the game is played in rounds; each round results in a transition being taken. A round is a sequential protocol, in which both players indicate their intended transition, and the transition taken is the one which occurs earlier.

In this section, we refer to an arbitrary hybrid automaton \mathcal{H} and its doubly-weighted labelled transition system $\mathcal{T}_{\mathcal{H}}$. Its exact definition is not important, and therefore it will not be formally introduced.

Turn-based games. Turn based game are played on a game graph defined over the transition system of the hybrid automaton. The game graph is induced by a partition of the set of control locations between the two players, Min and Max.

Given a partition of the set of control locations $\mathbf{L} = \mathbf{L}^{\text{Min}} \cup \mathbf{L}^{\text{Max}}$ of the hybrid automaton \mathcal{H} and the transition system $\mathcal{T}_{\mathcal{H}}$, we define the game graph $\Gamma_{\mathcal{H}} = \langle \mathcal{T}_{\mathcal{H}}, \mathbf{S}_{\mathcal{H}}^{\text{Min}}, \mathbf{S}_{\mathcal{H}}^{\text{Max}} \rangle$ as follows:

- $\mathbf{S}^{\text{Min}} = \mathbf{S} \cap (\mathbf{L}^{\text{Min}} \times \mathbb{R}^n)$, and
- $\mathbf{S}^{\text{Max}} = \mathbf{S} \cap (\mathbf{L}^{\text{Max}} \times \mathbb{R}^n) = \mathbf{S} \setminus \mathbf{S}^{\text{Min}}.$

By providing a payoff function \mathbf{P} , we obtain an actual game.

Quasi-concurrent games. Turn based games do not model the concurrent interaction between the controller and the environment that we would like to capture. We are looking for means to model the following type of interaction: both players, independently, choose how long to stay in the current control location and what control action to execute afterwards; the transition system changes its state according to the choice with a shorter stay. Quasi-concurrent games are a generalisation of turn based games and are a broadly used attempt at more accurate

modelling of the concurrent interaction between the controller and the environment [BBC06; BBC07; BBC09; BBJ⁺08; JLR09; BBC10]. We use the qualifier “quasi” because, although we are modelling concurrent interaction, the approach is in fact sequential.

To formally define the quasi-concurrent approach we need the notion a *timed action*. A timed action (a, t) , an element of $A \times \mathbb{R}_{\geq 0}$, is admissible from state s iff $s \xrightarrow{t} s'' \xrightarrow{a} s'$ is a finite run of the transition system. We will write $s \xrightarrow{a}_t s'$ for short.

A quasi-concurrent game on a hybrid automaton is given by a partition of the A set into the sets A^{Min} and A^{Max} . This models the fact that players are in control of control actions, rather than the states of the automaton. Let s be the current state of the transition system. A quasi-concurrent game is played in rounds. In every round, the players proceed as follows:

1. player Min, who models the controller, chooses an admissible timed action (a, t) such that $a \in A^{\text{Min}}$;
2. player Max, who models the environment, chooses an admissible timed action (a', t') such that $(a', t') = (a, t)$, or $a \in A^{\text{Max}}$ and $t' \leq t$;
3. player Max chooses a state s' such that $s \xrightarrow{a'}_{t'} s'$; state s' becomes the current state.

In concurrent interaction, both players make their time-action choices independently, whereas, in the quasi-concurrent setting, the choices are made sequentially. In effect, player Max has perfect information on the intentions of player Min, and cannot be surprised by his choice of a timed action. If we choose to view the choice of the systems next behaviour as a game between the two players, then the quasi-concurrent approach is similar in spirit to the upper value analysis for a game in strategic form (recall Section 2.2.1). In a truly concurrent setting the players' interaction resembles a game in strategic form, i.e., players make their choices simultaneously and independently. In this context issues such as preventing Zeno

behaviour arise, and although there are methods of alleviating this problem, they come at a cost of uniform determinacy [dAFH⁺03].

Notice that the quasi-concurrent game is not played on a game graph, as was the case for turn-based games. However, as it will be explained in Chapter 5, one can construct a special game graph, and define a turn-based game on this graph, that simulates the quasi-concurrent game as defined here.

Remark 2.41. *Notice that in the quasi-concurrent game setting we are implicitly restricting the semantics of the hybrid automaton. Admissible executions are in fact alternating sequences of continuous and discrete transitions. This change of semantics is a consequence of using the notion of a timed action.* \square

The following example illustrates the comment made in Remark 2.41.

Example 2.42. *Consider the diagram a) in Figure 2.8. The two functions, $f_1, f_2 : \mathbb{T} \rightarrow \mathbb{R}$ model the admissible flows in a location ℓ . Assume there is a control action a admissible from every state (ℓ, x) , for $x \in \mathbb{R}^n$. The following two are possible executions of the hybrid automaton from the state (ℓ, y_3) :*

$$(\ell, y_3) \xrightarrow{x_1} (\ell, y_1) \xrightarrow{x_2 - x_1} (\ell, y_2) \xrightarrow{a} \dots$$

and

$$(\ell, y_3) \xrightarrow{x_1} (\ell, y_1) \xrightarrow{x_2 - x_1} (\ell, y_0) \xrightarrow{a} \dots$$

In timed action semantics, there would be only one possible kind of execution:

$$(\ell, y_3) \xrightarrow{t} (\ell, f_1(t)) \xrightarrow{a} \dots$$

In particular, with timed action semantics in place, the state (ℓ, y_2) would not be reachable from (ℓ, y_2) through continuous transitions only.

The flow predicate can be viewed as a specification of admissible trajectories for the evolution of the values of the variables during a continuous transition. In a

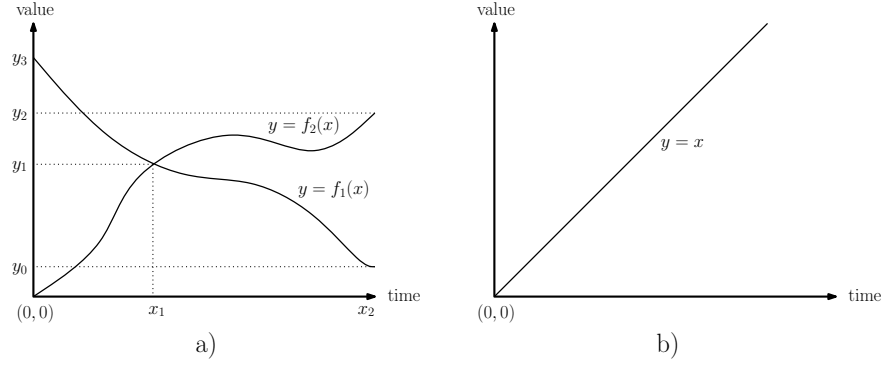


Figure 2.8: Hybrid automaton with multiple trajectories (diagram a)) and with a single trajectory (diagram b)).

single continuous transition the evolution follows a single trajectory. In timed action semantics there is exactly one continuous transition between two consecutive discrete transitions. In the general semantics, there may be several continuous transitions between two consecutive discrete transitions, and for each of these transitions the evolution of continuous variables may follow a different trajectory — this affects the set of reachable states. When the flow predicate specifies only one trajectory then the timed action semantics is not restrictive. This is the case for timed automata. Diagram b) in Figure 2.8 shows the admissible trajectories of timed automata. \square

Chapter 3

Games on single-clock timed automata

Timed automata are a subclass of hybrid automata with strong restrictions on admissible continuous changes to the values of the real-valued variables. As we have explained in Sections 2.3 and 2.3.2, in timed automata the values of the variables change at a constant and uniform rate and they are commonly referred to as clocks. In this chapter we will focus on timed automata restricted to a single clock only, and referred to as *single-clock timed automata*. The main goal of this chapter is to show that reachability-price games on single-clock timed automata are positionally determined, and that they are decidable in EXPTIME.

We start this chapter by defining the notion of a reachability-price game on a single clock-timed automaton and the notion of a cost function (Section 3.1). We adopt the approach introduced in Section 2.3.3. However, we introduce some technical modifications which will aid presentation — namely, aided by the notion of a cost function, the modifications will simplify dividing the problem of computing the game value into simpler subproblems. We conclude by discussing simplifying restrictions to the model (Section 3.1.1), and some basic algebraic operations, which we will be applying to the simplified single-clock timed automata (Section 3.1.2).

The key result is that by restricting our consideration to the simplified single-clock timed automata we do not lose generality (Theorem 3.10).

In Section 3.2 we study the properties of cost functions. Cost functions are piecewise-affine functions that are continuous and non-increasing. These functions naturally arise when one tries to compute the game value of a reachability-price game on a single-clock timed automaton. The theory of cost functions, which we develop in Section 3.2, will be instrumental in establishing the main results of this chapter.

We conclude this chapter with the results section (Section 3.3). First we present an EXPTIME algorithm for deciding the value of a reachability-price game on a single-clock timed automaton (Section 3.3.1). This algorithm extends the work of Bouyer et al. [BBM06], who have provided the first algorithm for this problem that was argued to run in triply-exponential time. In Section 3.3.1 we prove that reachability-price games on single-clock timed automata are positionally determined, and then, in Section 3.3.2, we establish that the algorithm is correct and that its running time is at most exponential.

3.1 Timed automata and reachability-price games

Our goal is to define the notion of turn-based reachability-price games on single-clock timed automata. In short, this will be done by defining a game on the transition system of the automaton. The main problem studied in this chapter is the complexity of *deciding the value of the reachability-price game* on a single-clock timed automaton (see Definition 2.17), which can be stated as follows:

Given a reachability-price game on a single-clock timed automaton, a state of the automaton, and a constant c , decide whether the value of the game from the given state is at most c .

Reachability-price games on transition systems. For technical reasons, our definition of a reachability-price game differs slightly from the one found in Chapter 2 (see Section 2.2.2). In the definition presented in this chapter, entering a goal state comes at an additional price, not related to the price of the transition taken. This extension is helpful in the following situation. Suppose the value of the reachability-price has been computed for some subset of states; we can then modify the game so that those states become goal states, with the entry price equal to the game value of that state, and compute the game value for the remaining states, by computing it in the modified game. We will use this approach, both in the algorithm, and in the proofs.

We now define reachability-price games on a transition system. The approach is similar to that presented in Section 2.2.2, however, we account for prices in goal states.

In order to define a reachability-price game, $\langle \Gamma, F, \pi^F \rangle$, we need to provide a weighted game graph Γ (see Definition 2.14), a set of designated goal states $F \subseteq S$, and *goal price function* $\pi^F : F \rightarrow \mathbb{R}_{\geq 0}$, an assignment of prices to goal states.

We now define the modified reachability-price payoff function \mathcal{P} , which accounts for prices in goal states. Given a reachability price game Γ and a run ω , we define \mathcal{P} as follows:

$$\mathcal{P}(\omega) = \begin{cases} \pi^F(\text{Last}(\omega_{\text{Stop}(\omega)})) + \text{Price}(\omega_{\text{Stop}(\omega)}) & \text{if } \text{Stop}(\omega) < \infty, \\ \infty & \text{otherwise.} \end{cases}$$

The definitions of the Price and Stop functions can be found in Section 2.2.2.

Cost functions. We now introduce the notions of a cost function and a quasi-cost function that will be used to define reachability-price games on timed automata. A reachability-price game on a timed automaton will be defined as a reachability-price game on its transition system. An assignment of quasi-cost functions to goal

locations of the automaton will be used to define the goal price function in the transition system.

We will be dealing with the ordered set of real numbers augmented with the greatest element, positive infinity denoted by ∞ . For that purpose we need to extend the $+$, \min and \max operators in a natural way. For $a \in \mathbb{R} \cup \{\infty\}$ we have $a + \infty = \infty$, $\max(a, \infty) = \infty$ and $\min(a, \infty) = a$.

We now introduce the definition of a cost function.

Definition 3.1 (Cost function). *A function $f : I \rightarrow \mathbb{R} \cup \{\infty\}$ that is continuous, non-increasing, and piecewise-affine, where I is a bounded interval, is said to be a cost function. We will write $\mathcal{CF}(I) \subseteq [I \rightarrow \mathbb{R} \cup \{\infty\}]$ to denote the set of all cost functions with the domain I .*

Remark 3.2. *Notice that if $f \in \mathcal{CF}(I)$, and $f(x) = \infty$ for some $x \in I$, then $f \equiv \infty$ over I .* \square

To define reachability-price games on single-clock timed automata we will need to weaken the notion of a cost function. Thus, we define the notion of a quasi-cost function.

Definition 3.3 (Quasi-cost function). *Given an interval $I = [0, M]$, where $M \in \mathbb{N}$, a function $f : [0, M] \rightarrow \mathbb{R}$ is said to be a quasi-cost function if f restricted to the interval $(i, i + 1)$ is a cost function, for every integer $i \in \{0, M - 1\}$. We will write $\mathcal{QCF}(I) \subseteq [I \rightarrow \mathbb{R} \cup \{\infty\}]$ to denote the set of all quasi-cost functions with the domain I .*

Reachability-price games on single-clock timed automata. We will be considering turn-based reachability-price games on single-clock timed automata. For the definition of a reachability-price game see Section 2.2.2, and for the definitions of timed automata and turn-based games see Section 2.3. As explained in Section 2.3.3, the game on a timed automaton will in fact be played on its transition

system, and the partition of states between players will be induced by a partition of the set of control locations. The definition of the reachability-price game, presented here, accounts for the prices in goal states.

As the name suggests, single-clock timed automata are a special case of timed automata, where the set of real-valued variables, i.e., clocks is a singleton set. This class is of special interest when considered in the context of reachability-price games. Determining the existence of optimal strategies in reachability-price games on timed automata with at least three clocks was shown to be undecidable by Bouyer et al. [BBM06], whereas for single-clock timed automata, a triply-exponential time algorithm was proposed by Bouyer et al. [BLMR06]. The decidability of this problem for two-clock timed automata remains open. In this chapter we extend the work initiated by Bouyer et al., and present an exponential time algorithm.

In Section 2.2.2 we have mentioned that timed automata yield deterministic transition systems. Indeed, a move of a player consists of either: choosing to delay for a certain time $t \in \mathbb{T}$, which corresponds to choosing a continuous transition labelled with t , or to change the control location, which corresponds to choosing a discrete transition labelled with the chosen control action $a \in \mathbf{A}$. In every state both choices have a unique outcome so, the transition system is indeed deterministic. This enables us to use the framework introduced in Section 2.2.2.

We will place the following assumption on the single-clock timed automata considered.

Assumption 3.4. *We assume (without loss of generality [BFH⁺01]) that \mathcal{A} is clock-bounded, i.e., there exists a positive constant M such that the $\text{inv}(l)[X := x]$ predicate is true only if $x \leq M$, for every control location l . The variable x denotes the single clock of the automaton* □.

Remark 3.5. *In a bounded single-clock timed automaton a clock constraint specifies a bounded interval, a subset of the set of non-negative reals. Throughout this chapter we will use such intervals to denote clock constraints.* □

To define a reachability-price game on a timed automaton \mathcal{A} , we use the game graph $\Gamma_{\mathcal{A}}$ (see Section 2.3.3) and use the reachability-price payoff function, as defined earlier in this section. To complete the definition, we need to specify the set of goal states, and a goal price function, which assigns price to the goal states. We will define the set of goal states by specifying a set of goal locations L^{F} . The goal price function, on the other hand, will be given in terms of quasi-cost function:

$$\mathsf{CF}^{\mathsf{F}} : \mathsf{L}^{\mathsf{F}} \rightarrow [[0, M] \rightarrow \mathbb{R} \cup \{\infty\}],$$

where M is the clock-bound mentioned in Assumption 3.4.

The following definition formalises the notion of a reachability-price game on a single-clock timed automaton.

Definition 3.6 (Reachability-price game). *Given a single-clock timed automaton \mathcal{A} , a partition of its set of control locations $\mathsf{L} = \mathsf{L}^{\mathsf{Min}} \cup \mathsf{L}^{\mathsf{Max}}$, a designated set of goal locations $\mathsf{L}^{\mathsf{F}} \subseteq \mathsf{L}$, and a function $\mathsf{CF}^{\mathsf{F}} : \mathsf{L}^{\mathsf{F}} \rightarrow \mathcal{QCF}([0, M])$, which assigns a quasi-cost function to every goal location, the reachability-price game on a single-clock timed automaton is given by a reachability-price game $\langle \Gamma_{\mathcal{A}}, \mathsf{F}, \pi^{\mathsf{F}} \rangle$ on the transition system $\mathcal{T}_{\mathcal{A}}$ where:*

- $\Gamma_{\mathcal{A}}$ is the weighted game graph,
- $\mathsf{F} = (\mathsf{L}^{\mathsf{F}} \times \mathcal{V}) \cap \mathsf{S}_{\mathcal{A}}$ is the set of goal states, and
- $\pi^{\mathsf{F}} : \mathsf{F} \rightarrow \mathbb{R}_{\geq 0}$ is the goal price function defined

$$\pi^{\mathsf{F}}((\ell, x)) = \mathsf{CF}^{\mathsf{F}}(\ell)(x),$$

for $s = (\ell, x) \in \mathsf{F}$.

We use the reachability-price payoff function \mathcal{P} as defined earlier in this section.

The size of the automaton, denoted by $|\mathcal{A}|$, is the total number of bits needed to represent all of its components. The size of the reachability-price game Γ , denoted by $|\Gamma|$, is equal to the size of the automaton \mathcal{A} and the total number of bits necessary to encode the remaining components. All constants are encoded in binary.

Remark 3.7. *Game graphs, as considered in this chapter, admit only positive prices on transitions. As a consequence, both the lower and the upper value of a reachability-price game are bounded by zero from below.* \square

3.1.1 Simplifications

In this section we are going to place some simplifying restrictions on the structure of timed automata, which will allow us to concentrate on the essence of the problem. We will also explain why these restrictions can be made without losing generality. In the remainder of this chapter we will be assuming that we are dealing with the restricted timed automata.

The restrictions and simplifications discussed in this section are not new, and were considered previously by Bouyer et al. in their work regarding reachability-price games on single-clock timed automata [BLMR06]. In particular, their complexity results are stated only for the restricted automata. The constructions, which allow us to restrict our considerations to the class of automata that do not have clock-resetting control actions, and whose control actions have zero prices, are the same as in [BLMR06]. We include them, and sketch the proof of their correctness, for the sake of completeness. The constructive proof that restricting to $[0, 1]$ -bounded single-clock timed automata can be done without losing generality is different from the one found in [BLMR06], as it does not involve an exponential blowup in the size of the automaton. The proof is new, but it bears similarities with the construction presented in the work of Laroussinie et al. [LMS04], with the only difference that our construction accounts for prices.

We start by defining the notion of a simple timed automaton, then proceed

to prove that by restricting to this model we do not lose generality, and conclude this section, with a discussion on syntactical simplifications.

Consider an interval I . We will write \mathcal{A}_I , for some I -bounded timed automaton, i.e., an automaton whose transition system has the state space restricted to $\mathsf{L} \times I$, and for every ℓ , the invariant is $\text{inv}(\ell) \cap I$.

Definition 3.8 (Simple timed automata). *We say that a single-clock timed automaton \mathcal{A} is simple if it is $[0, 1]$ -bounded, for every control action the resetting set is empty, and the price is zero, i.e., for every $a = (\ell, g, Z, \ell') \in \mathsf{A}$ we have $Z = \emptyset$, and $\pi(a) = 0$.*

The most important property of simple timed automata is as follows.

Remark 3.9. *In simple timed-automata time always progresses, i.e., if $(\ell, x) \xrightarrow{\sigma} (\ell', x')$ is a transition in a simple timed automaton's transition system, then $x' \geq x$. We will call this property the time progression property.* \square

We have the following result regarding simple timed automata.

Theorem 3.10. *The problem of deciding the value of a reachability-price game on single-clock timed automata is exponential time Turing-reducible to the analogous problem on simple single-clock timed automata.*

Theorem 3.10, restricted to $[0, 1]$ -bounded single-clock timed automata, was first proved by Bouyer et al. [BLMR06]. One can extend their proof to the general single-clock setting by using the polynomial encoding of bounded single-clock timed automata using $[0, 1]$ -bounded single-clock timed automata proposed Laroussinie et al. [LMS04], which can be easily adapted to the weighted setting. The detailed proof of Theorem 3.10 can be found in Appendix A.

To further simplify the model, we assume that the simple timed automaton is defined using closed clock constraints only. Once again, we make this assumptions to simplify our reasoning, but it is done without loss of generality. Theorem 3.25

can be easily adapted to produce ε -optimal strategies that respect open guards (it would involve a construction similar to the one found in the proof of Theorem 5.14). This is possible because the Val function, as computed by the algorithm presented in Section 3.3, is uniformly continuous over the interval $[0, 1]$, in every control location.

Simple timed automata, defined using closed clock constraints, admit three possible control action guards, namely $[0, 0]$, $[1, 1]$, and $[0, 1]$. The first kind does not allow for a continuous transition prior to the discrete one, and it is satisfied only by finitely many states. As it will be visible in the proofs of Section 3.3, the value of the Val function for such states, due to the time progression property of simple timed automata, does not “affect” the values for the other states. Transitions with guards of this kind can be dealt with, in polynomial time, during post-processing. The effect of a discrete transition, featuring a guard of the second kind, can be encoded using additional goal cost functions. Once again, proofs in Section 3.3 explain how this can be done. It is only the third kind of guards that can not be dealt with by such simple means. In light of this, and to simplify the presentation, we assume that all transition guards are true. A similar approach was used in the work of Bouyer et al. [BLMR06].

Remark 3.11. *In the light of the assumptions made, it is natural to think of the simple timed automaton, as given by $\mathcal{A} = \langle \mathbf{X}, \mathbf{L}, \mathbf{A}, \text{urg}, \pi \rangle$, where*

- $\mathbf{X} = \{x\}$ *is the set containing a single real-valued variable, referred to as the clock,*
- \mathbf{L} *is the finite set of control locations,*
- $\mathbf{A} \subseteq \mathbf{L} \times \mathbf{L}$ *is the finite set of control actions,*
- $\text{urg} : \mathbf{L} \rightarrow \{0, 1\}$ *is the urgency mapping,*
- $\pi : \mathbf{L} \rightarrow \mathbb{N}$ *is the price function.*

Notice that in this definition a control action (ℓ, ℓ') corresponds to $(\ell, \text{true}, \emptyset, \ell')$ in the original definition.

Additionally, notice that in the definition of a reachability-price game on a simple timed-automaton, the function CF^F assigns cost functions to locations (rather than quasi-cost functions). \square

Assumption 3.12. We assume that the upper value of the reachability-price game on a simple timed automaton is finite. \square

In light of Remark 3.11, the Assumption 3.12 can be made without loss of generality; one can determine the set of states from which player Max can prevent reaching goal by determining the appropriate set of control locations, which can be done in polynomial time. The real complexity lies in determining the value of a reachability-price game, given that player Min can ensure it.

3.1.2 Operations

We will now define some simple algebraic operations that we will be performing on cost functions and reachability-price games on simple timed automata. These operations will be used in the algorithm, presented in Section 3.3.1.

The first operation is the *override operation* [BJSV10]. This operation allows to combine two functions, with possibly intersecting domains, into a new function, whose domain is the union of the domains of the two original functions. The name stems from the fact, that the first of the two functions overrides the values of the second, i.e., the new function behaves like the first, on the domains' intersection. Formally, given two functions $h : I_1 \rightarrow \mathbb{R}$ and $g : I_2 \rightarrow \mathbb{R}$, we will write $f \triangleright g$ to denote the override operation on these two functions, defined as

$$(h \triangleright g)(x) = \begin{cases} h(x) & \text{if } x \in I_1 \\ g(x) & \text{if } x \in I_2 \setminus I_1 \end{cases}$$

As we have explained earlier, the algorithm for computing the game value will be in part iterative, i.e., it will iteratively compute the value of the game over a certain interval. We will use the override operation, to combine the result of two subsequent iterations.

We will now introduce three operations on reachability-price games on simple timed automata. For the purpose of their definitions, we fix an interval $I \subseteq [0, 1]$, an automaton \mathcal{A} , and a reachability-price game Γ on \mathcal{A} . As a result of each of the three operations we will obtain a game reachability-price Γ' on an automaton \mathcal{A}' ; the interval I is unchanged. As it will be explained in Section 3.3, during its execution, the algorithm for computing the game value will often restrict the game to intervals $I \subseteq [0, 1]$ — that is why, in the following definitions, we clearly indicate the interval.

$\Gamma[\text{urg}(\ell) := 1]$ denotes the game Γ' obtained from Γ by modifying the urgency mapping of the automaton \mathcal{A} so that ℓ becomes an urgent control location, i.e., in the automaton \mathcal{A}' we have:

$$\text{urg}'(\ell') = \begin{cases} 1 & \text{if } \ell' = \ell \\ \text{urg}(\ell) & \text{otherwise,} \end{cases}$$

and the remaining components remain unchanged. As this operation does not alter the set of control locations, the partition of the set of control locations, the set of goal locations, and the assignment of cost functions to goal locations, remain unchanged in Γ' .

$\Gamma[\mathbf{L}^F \cup \ell, h]$ denotes the game obtained from Γ by adding ℓ to the set of goal locations, with h being the cost function assigned to ℓ . Formally, to obtain the game Γ' we set $\mathbf{L}^{F'} = \mathbf{L}^F \cup \{\ell\}$, and define the mapping from goal locations to cost functions as:

$$\text{CF}^{F'}(\ell') = \begin{cases} h & \text{if } \ell' = \ell, \\ \text{CF}^F(\ell') & \text{if } \ell' \in \mathbf{L}^F. \end{cases}$$

We do not require that $\ell \in \mathbf{L}$, i.e., ℓ can be a fresh control location, however, it is required that $h \in \mathcal{CF}(I)$. If $\ell \in \mathbf{L}$ then the automaton \mathcal{A} and the partition of its control locations between players remain unchanged — all components of \mathcal{A}' and Γ' have been defined. If, however, $\ell \notin \mathbf{L}$ then we complete the definition of the automaton \mathcal{A}' in the following way:

$$\text{urg}'(\ell') = \begin{cases} 1 & \ell' = \text{if } \ell, \\ \text{urg}(\ell') & \text{otherwise,} \end{cases} \quad \text{and } \pi'(\ell') = \begin{cases} 0 & \ell' = \text{if } \ell, \\ \pi(\ell') & \text{otherwise,} \end{cases}$$

and the definition of the reachability-price Γ' , by setting $\mathbf{L}^{\text{Max}'} = \mathbf{L}^{\text{Max}} \cup \{\ell\}$.

$\Gamma[\mathbf{A} \cup a]$ denotes the game obtained from Γ by adding an additional control action $a \in \mathbf{L} \times \mathbf{L}$ in the automaton \mathcal{A} . We set $\mathbf{A}' = \mathbf{A} \cup \{a\}$; the remaining components of the automaton remain unchanged. This operation does not affect any other components of the reachability-price game.

We now briefly explain the applications of the above-mentioned operations. In the algorithm for computing the game value, with each recursive call a non-urgent control location is removed, and the algorithm proceeds to compute the game value for the new game. Removing a control location of player Max consists of changing its urgency mapping, so that it becomes urgent — the first operation is being used. On the other hand, removing a control location of player Min consists of turning it into a goal location — the second operation is being used. Finally, as we have explained in the context of the override operation, apart from being recursive, the algorithm for computing the game value contains an iterative component. We use the third operation, together with the override operation, to combine the results of subsequent iterations.

3.2 Cost functions

In this section we state and prove the key properties of cost functions, which were introduced in Section 3.1. Cost functions are a central notion when considering reachability-price games on single-clock timed automata. The theory of cost functions will be used to construct the algorithm for computing the Val function for reachability-price games on simple timed automata and to prove its correctness.

The main result of this section is Proposition 3.18, which establishes the key properties of the minC and maxC operators, introduced in this section, that transform cost functions. This result will be essential in establishing the proofs of the results presented in Section 3.3.

Recall Definition 3.1, which introduced the notion of a cost function. At times, we will need to talk about pieces of a cost function, i.e, individual affine functions. To make this easier, we introduce the following convention. Given an interval I and a cost function $f : I \rightarrow \mathbb{R}$, we will write $f = \langle f_1, \dots, f_k \rangle$ to denote the fact that the piecewise-affine function f consists of affine pieces f_1, \dots, f_k , with domains I_1^f, \dots, I_k^f , where k is the smallest integer such that

$$f(x) = \begin{cases} f_1(x) & \text{if } x \in I_1^f \\ f_2(x) & \text{if } x \in I_2^f \\ \dots & \\ f_k(x) & \text{if } x \in I_k^f. \end{cases}$$

Throughout the chapter, we will be implicitly assuming that $I = [b, e]$, and that $I_i^f = [b_i^f, e_i^f]$, for $i \in \{1, \dots, k\}$, with $e_{i+1}^f = b_i^f$, for $i \in \{1, \dots, k-1\}$. The formula for the individual segment f_i will be given by $a_i^f \cdot x + c_i^f$, for $i \in \{1, \dots, k\}$. If f is clear from the context, we will omit the superscript.

Remark 3.13. Notice that a cost function, $f = \langle f_1, \dots, f_k \rangle$, such that I_i admits

rational end-points, and that a_i^f and b_i^f are rational, for all $i \in \{1, \dots, k\}$ is definable.

Fact 3.14. *If $f, g : I \rightarrow \mathbb{R} \cup \{\infty\}$ are cost functions, then their pointwise minimum and maximum, i.e., functions defined as*

$$x \mapsto \min\{f(x), g(x)\} \text{ and } x \mapsto \max\{f(x), g(x)\},$$

are cost functions as well.

Proof. If one of the functions is equivalent to ∞ over I , then the fact is trivially true. Otherwise, let m be the total number of pieces in both functions. As each piece is an affine function, there can be at most m^2 intersections. Each function is piecewise-affine between the two possible subsequent intersections. This assures both continuity and piecewise-affinity of the pointwise minimum and maximum.

To conclude the proof, we argue that both the pointwise minimum and maximum are non-increasing. As there are finitely many intersections, we can finitely partition I in such a way that on every element of the partition, we have $f \leq g$ or $f \geq g$. Hence, on every element of the partition, the pointwise minimum and maximum are non-increasing. This together with the fact that we have continuity finishes the proof. \square

Fact 3.15. *Let $S(n, m)$ denote the maximum number of affine pieces in a pointwise minimum (maximum) of two piecewise-affine functions with m and n pieces respectively. We have that $S(n, m) \leq 2(n + m - 1)$.*

Proof. We clearly have the following $S(1, 1) \leq 2$ and $S(n, m) = S(m, n)$. First we will prove that $S(n, 1) \leq 2n$, for $n > 0$. Assume that it is true for n . If we have $n + 1$ pieces, the pointwise minimum (maximum) may split the $n + 1$ -th piece into at most two pieces, so we get $S(n + 1, 1) \leq S(n, 1) + 2 \leq 2n + 2$, which is the desired result.

Now we consider the remaining case, i.e., $n, m > 1$. If we assume that

$S(n, m) \leq 2(n + m - 1)$, then to complete the proof it is sufficient to show that $S(n + 1, m) = 2(n + m)$. Let us assume that the first n pieces intersect first $k \leq m$ pieces. In that case, the new piece, can intersect at most $m - k + 1$ pieces. So the maximum number of pieces is $S(n, k) + S(1, m - k + 1)$. Using inductive hypothesis we get $S(n + 1, m) \leq 2(n + k - 1) + 2(m - k + 1) \leq 2(n + m)$. \square

We now introduce two operators, which are key in defining the relationship between game values in adjacent control locations. This will be later summarised by Lemma 3.26. Given a function $f : [b, e] \rightarrow \mathbb{R} \cup \{\infty\}$ and a positive constant c , we define the following two operators that create new functions, of the same type as the function f :

$$\begin{aligned} \min C(f, c) &: x \mapsto \min_{0 \leq t \leq e-x} c \cdot t + f(x + t) \\ \max C(f, c) &: x \mapsto \max_{0 \leq t \leq e-x} c \cdot t + f(x + t). \end{aligned}$$

The following example illustrates the intuition behind the $\min C$ and $\max C$ operators.

Example 3.16. Figure 3.1 gives an intuitive understanding of the $\min C(f, c)$ operator, for a cost function f and a positive real constant c . The cost function is given as $f = \langle f_1, \dots, f_7 \rangle$ (as seen in Figure 3.1a). The slope of f_2 and f_6 is steeper than $-c$; for the remaining components it is shallower. The cost function $g = \min C(f, c)$ is depicted in Figure 3.1b), and is given by $\langle g_1, \dots, g_6 \rangle$. All components of g have a slope shallower than or equal to $-c$. The formula for g_1 is the same as for f_1 , however, the domain is a subset (similarly for f_4 and g_4). The function g_3 is equal to f_3 (similarly g_6 is equal to f_7). Functions g_2 and g_5 , have the slope $-c$, and were not present in f .

Similarly, Figure 3.2 illustrates the intuition behind the $\max C$ operator. Cost function f is given as previously (see Figure 3.2a). The slopes of f_1 , f_3 , and f_7 are shallower than $-c$; for the remaining components it is steeper. The cost function $h =$

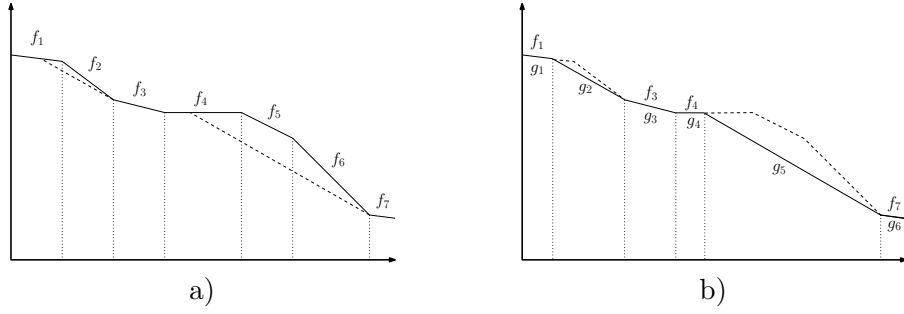


Figure 3.1: a) Cost function f , before applying $\min C(f, c)$ operator — dashed lines have a slope $-c$; b) cost function $g = \min C(f, c)$ — dashed lines denote parts of f that do not coincide with g .

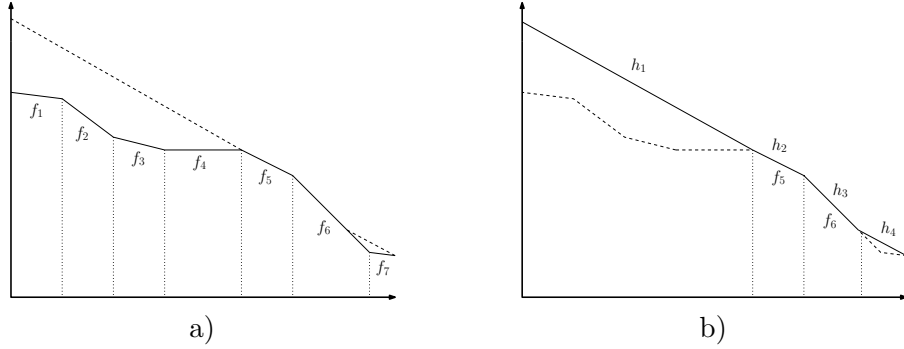


Figure 3.2: a) Cost function f , before applying $\max C(f, c)$ operator — dashed lines have a slope $-c$; b) cost function $h = \max C(f, c)$ — dashed lines denote parts of f that do not coincide with h .

$\max C(f, c)$ is depicted in Figure 3.2b), and is given by $\langle h_1, \dots, h_4 \rangle$. All components of h have a slope steeper than or equal to $-c$. The formula for h_2 and h_3 is the same as that of f_5 and f_6 , respectively. Moreover, the domain of h_2 coincides with the domain of f_5 , however, the domain of h_3 is a strict subset of the domain of f_6 . Functions h_1 and h_4 have the slope $-c$, and were not present in f . \square

We now proceed to state and prove the properties of the $\min C$ and $\max C$ operators when applied to cost functions. The key property is that an application of either of those operators to a cost function is also a cost function.

Lemma 3.17. *Let c be a positive constant. If $f : [b, e] \rightarrow \mathbb{R}$ is a cost function then*

$\min C(f, c)$ is a cost functions as well. The same holds for $\max C$.

Proof. If f is equivalent to ∞ , over $[b, e]$ then the lemma is trivially true. Therefore, we concentrate on the remaining case. Clearly $\min C(f, c)$ is continuous. We fix $x_1 \leq x_2 \in [b, e]$, and for every $x \in [b, e]$ we will use the following notation:

$$t(x) = \max\{t^* : t^* = \arg \min_{0 \leq t \leq e-x} c \cdot t + f(x+t)\}.$$

To show piecewise-affinity observe the following, if $t(x_1) > 0$ is such that $x_1 + t(x_1) \geq x_2$, then $t(x_2) = t(x_1) - x_2 + x_1$. Moreover,

$$\min C(f, c)(x_2) = \min C(f, c)(x_1) - (x_2 - x_1) \cdot c,$$

i.e., $\min C(f, c)$ is linear with a slope $-c$ over the interval $[x_1, x_1 + t(x)]$. If, on the other hand, for all $x \in I \subseteq [b, e]$ we have $t(x) = 0$, then $\min C(f, c)(x)$ is equal to f_i on the interval $I \cap I_i$, and $a_i \geq -c$.

We will show that $\min C(f, c)$ is piecewise-affine in an inductive fashion. Recall that $f = \langle f_1, \dots, f_k \rangle$. Our inductive hypothesis is as follows: $\min C(f, c)$ is piecewise-affine on $[b, v]$, and $v = b_i$ for some $i \in 1, \dots, k$.

The hypothesis clearly holds for $v = b$, so let's assume $v > b$. Let $x \in [v, e]$ be smallest such that $t(x) > 0$. Clearly, $\min C(f, c)|_{[v, x]} = f|_{[v, x]}$, and it is affine on the interval $[x, x + t(x)]$ with a slope $-c$.

We have shown, that $\min C(f, c)|_{[v, x+t(x)]}$ is piecewise-affine. It remains to show that $x + t(x) = b_i$, for some $i \leq k$, or that $x + t(x) = e$. Let's assume otherwise, i.e., that $x + t(x) \in (b_i, e_i)$ for some $i \leq k$. We know that $t(b_i) = x + t(x) - b_i > 0$,

which implies that

$$\begin{aligned}
f(b_i) &\geq \min C(f, c)(b_i) \\
&= c \cdot t(b_i) + f(b_i + t(b_i)) \\
&= c \cdot t(b_i) + f_i(b_i + t(b_i)) \\
&= c \cdot t(b_i) + a_i^f t(b_i) + f(b_i),
\end{aligned}$$

which in turn implies that $a_i \leq -c$. If we take $t = e_i - x$, we will get

$$f(x + t(x)) \geq f(x + t).$$

This, however, contradicts the definition of $t(x)$, and hence $x + t(x) = b_i$ for some i , or $x + t(x) = e$. This finishes the proof of inductive step's correctness.

It takes at most i inductive steps to prove, that $\min C(f, c)$ is piecewise-affine on $[b, b_i]$, and i is bounded by k . This yields that $\min C(f, c)$ is piecewise-affine.

We will now proceed to show that $\min C(f, c)$ is non-increasing. Fix $x_1 \leq x_2 \in [b, e]$. We will show that for every $t(x_1)$, there exists a $t \in [0, e - x_2]$ such that

$$c \cdot t + f(x_2 + t) \leq c \cdot t(x_1) + f(x_1 + t(x_1)).$$

If $t(x_1) \in [0, x_2 - x_1]$, then $t = 0$. We have:

$$c \cdot t(x_1) + f(x_1 + t(x_1)) \geq f(x_1 + t(x_1)) \geq f(x_2).$$

If, on the other hand, $t(x_1) \in [x_2 - x_1, e - x_1]$, then $t = t(x_1) - x_2 + x_1$. We have:

$$\begin{aligned}
c \cdot t(x_1) + f(x_1 + t(x_1)) &\geq c(t(x_1) - x_2 + x_1) + f(x_1 + t(x_1)) \\
&= c(t(x_1) - x_2 + x_1) + f(x_2 + (t(x_1) - x_2 + x_1)).
\end{aligned}$$

This finishes the proof. \square

The following proposition formalises the intuition how the minC (maxC) operators affect cost functions. The minC (maxC) operator removes all pieces of a cost function that have slopes steeper (shallower) than $-c$, and substitutes them with pieces that have the slope equal to $-c$. For the remaining pieces, the formula remains unchanged, but the domain may change. However, the new domain is always a subset of the domain of the original domain. This property will be instrumental in proving that the algorithm, presented in Section 3.3.1 terminates.

Proposition 3.18. *Let $f = \langle f_1, \dots, f_k \rangle$ and let $\text{minC}(f, c) = \langle g_1, \dots, g_l \rangle$. We have that $l \leq k$, and $a_j^g \geq -c$, for all $j = 1, \dots, l$. Moreover, if $a_j^g > -c$ then there exists $i \in \{1, \dots, k\}$ such that the formulae of f_i and g_j are equal and $I_j^g \subseteq I_i^f \ni b_i^f$.*

For maxC we can prove a similar result, with the only difference that in the statement of Proposition 3.18, with the only difference that we have $a_j^g \leq -c$ and $a_j^g < -c$, respectively.

We conclude this section with the following result, which establishes that the order in which we take the pointwise minimum and apply the minC operator (similarly for pointwise maximum and maxC operator) does not matter.

Lemma 3.19. *If $f, g : [b, e] \rightarrow \mathbb{R} \cup \{\infty\}$ are cost functions, and c is a positive constant, then:*

$$\min\{\text{minC}(f, c), \text{minC}(g, c)\} = \text{minC}(\min\{f, g\}, c)$$

Similar equality holds if we use max and maxC operator.

Proof. First we treat the case, when one of the functions is equivalent to ∞ over

$[b, e]$. Without loss of generality we assume it is f . We then have:

$$\begin{aligned} \min\{\min C(f, c), \min C(g, c)\} &= \min\{f, \min C(g, c)\} \\ &= \min C(g, c) \\ &= \min C(\min\{g, f\}, c) \end{aligned}$$

The equalities are a consequence of how $+$, \min , and \max operators are defined over $\mathbb{R} \cup \{\infty\}$.

It remains to deal with the case where none of the functions is equivalent to ∞ . Fix some $x \in [b, e]$, and let $t(x)$ have the same meaning as in the proof of Lemma 3.17. We start by proving the \geq inequality. The right hand side is thus equal to:

$$c \cdot t(x) + \min\{f(x + t(x)), g(x + t(x))\}$$

Without loss of generality we can assume that this is in fact equal to $c \cdot t(x) + f(x + t(x))$. This, however, implies that

$$t(x) = \arg \min_{0 \leq t \leq e-x} c \cdot t(x) + f(x + t(x)),$$

and that in turn implies that $\min C(f, c)(x) \leq \min C(g, c)(x)$. This proves that the left hand side is not smaller than the right hand side. The proof of the \leq inequality is similar, and hence omitted. \square

3.3 Solving reachability-price games on simple timed automata

In this section we present an EXPTIME algorithm for computing the value of reachability-price games on single-clock timed automata, i.e., the main contribution of this chapter. The model and the games were introduced in Section 3.1,

and in Section 3.2 we have introduced the theory of cost functions, that will be instrumental in establishing the correctness and complexity of the algorithm. This work extends the work of Bouyer et al. [BLMR06], which established a 3EXPTIME algorithm for computing upper value of the reachability-price game.

The main results, presented in this chapter, are as follows.

Theorem 3.20. *Reachability-price games on simple timed-automata are positionally determined.*

Theorem 3.20 follows from the fact that the algorithm presented in this section is correct (Theorem 3.33), and from Theorem 3.25, which establishes the existence of positional almost optimal strategies.

Theorem 3.21. *The problem of deciding the value of a reachability-price game on simple timed automata is in EXPTIME.*

Theorem 3.21 follows from the fact that the algorithm, presented in this section, is correct (Theorem 3.33), and from Theorem 3.38, which establishes that the running time of the algorithm is at most exponential in the size of the input.

The following assumption underlies every proof and construction introduced in the remainder of this section. In Section 3.1.1 we have explained why this is without loss of generality.

Assumption 3.22. *In the remainder of this section we assume that the single-clock timed automaton, over which the reachability-price games is defined, is simple (as defined in Section 3.1.1).* □

The remainder of this section is organised as follows. We start this section by introducing an algorithm for computing the game value of reachability-price games on single-clock timed automata. We then show that correctness of this algorithm allows us to establish positional determinacy of the aforementioned games, and finally proceed to prove that the algorithm is indeed correct, and to that it works

in exponential time — EXPTIME. The theory of cost functions is an essential component of both the proof of correctness and of the complexity analysis.

In the following, we will be considering a game Γ and games Γ' and Γ'' derived from it. Furthermore, due to the iterative nature of our algorithm, we will often restrict the game to an interval $I \subseteq [0, 1]$. To ensure clarity, we will be writing Val_{Γ_I} to explicitly indicate the game Γ and the interval I (if it is of importance), to which the function refers. Unlike clock constraints, the interval I will usually have rational endpoints. This, however, does not increase complexity — the endpoints will naturally arise during the computation.

At times, it will be convenient to treat Val as an element of $[\mathbf{L} \rightarrow \mathcal{CF}(I)]$, rather than an element of $[\mathbf{S} \rightarrow \mathbb{R}]$. We will therefore abuse notation and write $\text{Val}(\ell)$ to denote the function $x \mapsto \text{Val}(\ell, x)$.

3.3.1 Computing the value of reachability-price games on single-clock timed automata

In this section we present the algorithm for computing the game value of reachability-price games on single-clock timed automata. This section consists of three parts. First, we introduce operations and constructions that will be used by the algorithm to combine the intermediate results of the computation. Second we introduce the procedure for computing the game value, and discuss how it differs from previously used approaches. We conclude this section with a result, which states that if the algorithm correctly computes the game value, then positional almost optimal strategies can be derived.

The algorithm presented in this section will work recursively with respect to the number of non-urgent control locations in the timed automaton, over which the reachability-price game is defined. With each recursive step, a game with one less non-urgent control location will be considered. This approach will be explained in full detail in the subsection in which the algorithm is defined. In order to make

handling of non-urgent control locations convenient, we introduce the following definition:

$$\text{NonUrgent}(\Gamma) = \{\ell : \ell \in \mathbf{L} \setminus \mathbf{L}^F \text{ and } \text{urg}(\ell) = 0\}.$$

We start by introducing an operation **CostConsistent**, that will be used to combine the results of subsequent iterations in the iterative component of the procedure **SolveRP**. Consider two intervals $I_1 = [b_1, e_1] \subseteq [0, 1]$ and $I_2 = [b_2, e_2] \subseteq [0, 1]$ such that $r = e_1 = b_2$, and some game Γ . We would like to have an operation, that allows us to compute $\text{Val}_{\Gamma_{I_1 \cup I_2}}$, i.e., the value of the game Γ restricted to the interval $I_1 \cup I_2$. Furthermore let's assume that we have computed $\text{Val}_{\Gamma_{I_2}}$, the value of the game Γ restricted to I_2 . The operation **CostConsistent** produces a new reachability-price game $\Gamma'_{I_1} = \text{CostConsistent}(\Gamma_{I_1}, \text{Val}_{\Gamma_{I_2}})$, with the following property, for every $\ell \in \mathbf{L}$:

$$\text{Val}_{\Gamma_{I_1 \cup I_2}}(\ell) = \text{Val}_{\Gamma'_{I_1}}(\ell) \triangleright \text{Val}_{\Gamma_{I_2}}(\ell).$$

The operation is defined as follows:

$$\begin{aligned} \text{CostConsistent}(\Gamma_{I_1}, \text{Val}_{\Gamma_{I_2}}) = \\ \left(\Gamma[\mathbf{L}^F \cup \ell'_1, x \mapsto \text{Val}_{\Gamma_{I_2}}(\ell_1, r) + (r - x)\pi(\ell_1)] [\mathbf{A} \cup (\ell_1, \ell'_1)] \dots \right. \\ \left. [\mathbf{L}^F \cup \ell'_k, x \mapsto \text{Val}_{\Gamma_{I_2}}(\ell_k, r) + (r - x)\pi(\ell_k)] [\mathbf{A} \cup (\ell_k, \ell'_k)] \right)_{I_1}, \end{aligned}$$

where $\{\ell_1, \dots, \ell_k\} = \text{NonUrgent}(\Gamma)$.

The intuition behind the **CostConsistent** operation is as follows. In Γ_{I_1} , the time can not progress past e_1 , whereas in $\Gamma_{I_1 \cup I_2}$ it can; this results in $\text{Val}_{\Gamma_{I_1}}$ being unrelated to $(\text{Val}_{\Gamma_{I_1 \cup I_2}})_{|\mathbf{L} \times I_1}$, although, due to the lack of resets, $\text{Val}_{\Gamma_{I_2}}$ is equal to $(\text{Val}_{\Gamma_{I_1 \cup I_2}})_{|\mathbf{L} \times I_2}$ — the time progression property. To alleviate this, for every non-urgent control location ℓ , we add a new goal location ℓ' whose cost functions encodes the following behaviour: upon entering ℓ wait till time e_1 , and then reach goal, from

the state (ℓ, e_1) , “optimally” as if Γ_{I_2} was the game being played.

We will also be considering situations where we have already computed $\text{Val}_\Gamma(\ell)$ (over some interval $I \subseteq [0, 1]$) for some control location ℓ of Γ . We will want to use this fact to compute Val_Γ for the remaining control locations. We will construct a game Γ' that has the following property

$$\text{Val}_{\Gamma'}(\ell', x) = \text{Val}_\Gamma(\ell', x),$$

for every control location $\ell' \in \mathbf{L}$ and every clock valuation $x \in I$. To obtain Γ' from Γ , we make ℓ a goal location. Consider $h : I \rightarrow \mathbb{R}_{\geq 0}$ defined as $h(x) = \text{Val}_\Gamma(\ell, x)$ for all $x \in I$. We define Γ' as

$$\Gamma' = \Gamma[\mathbf{L}^F \cup \ell, h].$$

Remark 3.23. *Notice that Γ' , as defined above, contains one less non-urgent control location than the original game Γ .* □

Algorithm

We will now introduce an algorithm, which given a reachability-price game on a single-clock timed automaton, computes the Val function, which to every state of the game graph, assigns the value of the game originating from that state. The use of word “computes” is meaningful because, as we will prove later, the Val function is locationwise a cost function and cost functions are definable (see Remark 3.13). The algorithm will be given in terms of a recursive procedure **SolveRP**.

The procedure **SolveRP** takes on input a reachability-price game Γ_I , where $I = [b, e] \subseteq [0, 1]$ and the automaton underlying Γ is simple. Upon termination, the function outputs the function Val_{Γ_I} , which is a cost function locationwise.

The algorithm works recursively with respect to the set of non-urgent control locations. During each recursive call, it identifies a non-urgent control location that minimises the value of the price function. There are two cases to consider, depending

on the ownership of the control location but both of them are handled in a similar fashion. The algorithm modifies the game Γ to have one less non-urgent control location. In case $\ell \in L^{\text{Max}}$, we convert ℓ to be urgent. If $\ell \in L^{\text{Min}}$, we convert ℓ to be a goal location that captures the following behaviour: once ℓ is reached in Γ , player Min spends all available time there. The intuition behind this is as follows: if $\ell \in L^{\text{Max}}$, it is unlikely that spending time in that control location will be beneficial for player Max. Likewise, when $\ell \in L^{\text{Min}}$, it is likely that it will be beneficial for player Min to stay as long as possible. There are cases, however, when this intuition is incorrect, i.e., it is beneficial, respectively, for player Max to wait, and for player Min to move immediately. This necessitates the iterative procedure, outlined in the following, employed during each recursive call.

The working assumption is that Val is a cost function locationwise. During a recursive call, for every control location the algorithm iteratively computes the result of the minC (maxC) operator applied to the minimum (maximum) of the cost functions in successor control locations (that are equal to Val). The iterative procedures in cases 2 and 3 of the algorithm compute the solution over a sequence of intervals, proceeding from right to left of the time axis. They first assume that the aforementioned intuition is correct (step 1), and then identify the rightmost interval, over which it is not (step 2). The next step is to adjust the solution over that interval (step 2 and 3). It remains to find the solution to the left of the found interval. This is done in the subsequent iterations.

We now present the recursive procedure $\text{SolveRP}(\Gamma_I)$.

First case: $\text{NonUrgent}(\Gamma_I) = \emptyset$.

$\text{Val}(\ell)$ is a cost function (for every control location ℓ) and can be computed by solving a finite game in polynomial time. If CF^F has p pieces in total, then Val has at most $2p$ pieces [BLMR06]. This is a consequence of Fact 3.15.

Second case: $L^{\text{Max}} \ni \ell^* = \arg \min \{\pi(\ell) : \ell \in \text{NonUrgent}(\Gamma)\}$.

In the second case of the algorithm, an iterative procedure is applied to

compute Val_Γ over the interval $I = [b, e]$; in each iteration, the computation is restricted to the interval $[b, r]$, with $r = e$ in the first iteration.

First, in Step 1, a game Γ' with one less non-urgent is constructed. We obtain Γ' from Γ by making ℓ^* an urgent control location — this captures the intuition that, since ℓ^* minimises the price function, it is beneficial for player Max to leave ℓ^* immediately.

Second, in Step 2, the procedure identifies the rightmost interval over which the function $f = \text{Val}_{\Gamma'}(\ell^*)$, computed in Step 1, has an affine piece with the slope strictly shallower than $-\pi(\ell^*)$; the affine piece and the interval are denoted by f_i and $[b_i, e_i]$, respectively. Third, in Step 2, a new game Γ'' is constructed; we are considering this game over the interval $[b_i, e_i]$. Like Γ' , the game Γ'' has one less non-urgent control location than the game Γ ; it is obtained from Γ by turning ℓ^* into a goal location with the cost function $h = -\pi(\ell^*)(r - x) + \text{Val}_{\Gamma_{[r, e]}}(\ell^*, r)$ assigned to ℓ^* — this cost function captures the behaviour contrary to the previously considered intuition, i.e., that, upon entering ℓ^* player Max spends all available time there. The game Γ'' is used to adjust the solution, to account for states from which the intuition that leaving ℓ^* immediately is beneficial for player Max is incorrect. The slope of f_i is shallower than $-\pi(\ell^*)$, and since ℓ^* minimises the price function, it means that f_i is actually an affine piece of one of the cost functions assigned to goal locations in the game Γ .

Finally, in Step 3 Val_Γ over $[b_i, e]$ is being established. It is equal to $\text{Val}_{\Gamma'}$ over the interval $[e_i, r]$ and to $\text{Val}_{\Gamma''}$ over the interval $[b_i, e_i]$. The algorithm then proceeds to the next iteration by setting $r = b_i$; the iterative procedure is completed when $b_i = b$. The **CostConsistent** operation is used to assure consistency of solutions between subsequent iterations.

The procedure is as follows:

1. Assuming that we have computed $\text{Val}_{\Gamma_{[r,e]}}$, for some $r \in I$, we set:

$$\Gamma'_{[b,r]} = \left(\text{CostConsistent} \left(\Gamma_{[b,r]}, \text{Val}_{\Gamma_{[r,e]}} \right) \right) [\text{urg}(\ell^*) := 1],$$

and compute $\text{Val}_{\Gamma'_{[b,r]}} = \text{SolveRP} \left(\Gamma'_{[b,r]} \right)$. Let $f = \langle f_1, \dots, f_k \rangle = \text{Val}_{\Gamma'_{[b,r]}}(\ell^*)$.

2. Let i be the smallest natural number such that $a_i^f > -\pi(\ell^*)$ and for all $j > i$ we have $a_j^f \leq -\pi(\ell^*)$. If $i > 0$, then we define $h : I_i^f \rightarrow \mathbb{R}$ as:

$$h(x) = -\pi(\ell^*)(e_i^f - x) + f_i(e_i^f),$$

and we define $\Gamma''_{[b_i^f, e_i^f]}$ as:

$$\Gamma''_{[b_i^f, e_i^f]} = \text{CostConsistent} \left(\Gamma'_{[b_i^f, e_i^f]}, \text{Val}_{\Gamma'_{[e_i^f, r]}} \right) [\mathbb{L}^F \cup \ell^*, h],$$

and compute $\text{Val}_{\Gamma_{[b_i^f, e_i^f]}} = \text{SolveRP} \left(\Gamma_{[b_i^f, e_i^f]} \right)$.

3. We set

$$\text{Val}_{\Gamma_{[b_i^f, e]}} = \text{Val}_{\Gamma''_{[b_i^f, e_i^f]}} \triangleright \text{Val}_{\Gamma'_{[e_i^f, r]}} \triangleright \text{Val}_{\Gamma_{[r,e]}}.$$

If $i = 0$ the Γ'' term is omitted.

We set $r = b_i^f$. If $r \neq b$ then goto 1, otherwise output Val_{Γ_I} .

We initialise the procedure by solving the game $\Gamma'_I = \Gamma_I[\text{urg}(\ell^*) := 1]$, and setting $r = e$. Observe that $\text{Val}_{\Gamma_{[e,e]}} = \text{Val}_{\Gamma'_{[e,e]}}$.

Third case: $\mathbb{L}^{\text{Min}} \ni \ell^* = \arg \min \{ \pi(\ell) : \ell \in \text{NonUrgent}(\Gamma) \}$.

In the third case of the algorithm, an iterative procedure is applied to compute Val_{Γ} over the interval $I = [b, e]$; in each iteration, the computation is restricted to the interval $[b, r]$, with $r = e$ in the first iteration.

First, in Step 1 a game Γ' with one less non-urgent is constructed. We obtain Γ' from Γ by making ℓ^* a goal location; the cost function h assigned to ℓ^*

captures the following behaviour: once ℓ^* is reached, player Min chooses to spend all available time there.

Second, in Step 2 the procedure identifies the rightmost interval over which the function f , computed in Step 1, is strictly smaller than h for at least one argument; the interval corresponds to an affine segment of f denoted by f_i . The argument, for which the functions f and h are equal, is denoted by $x^* \in I_i$ — such an argument always exists as $f(r) = h(r)$, by definition. Third, in Step 2 a new game Γ'' is constructed. Like Γ' , the game Γ'' is obtained from Γ by turning ℓ^* into a goal location. In this case, however, the cost function assigned to ℓ^* is f_i , and the game is being considered over the interval $[b_i, x^*]$ — the cost function f_i captures the intuition that it is beneficial to leave ℓ^* immediately. The game Γ'' is used to adjust the solution, to account for states from which the intuition that spending all available time in ℓ^* is beneficial for player Min is incorrect. The slope of f_i is shallower than that of h , which is equal to $-\pi(\ell^*)$, and since ℓ^* minimises the price function, it means that f_i is actually an affine piece of one of the cost functions assigned to goal locations in the game Γ .

Finally, in Step 3 Val_Γ over $[b_i, e]$ is being established. It is equal to $\text{Val}_{\Gamma'}$ over the interval $[x^*, r]$ and to $\text{Val}_{\Gamma''}$, over the interval $[b_i, x^*]$. The algorithm then proceeds to the next iteration by setting $r = b_i$; the iterative procedure is completed when $b_i = b$. The **CostConsistent** operation is used to assure consistency of solutions between subsequent iterations.

The procedure is as follows:

1. Assuming that we have computed $\text{Val}_{\Gamma_{[r,e]}}$, for some $r \in I$, let $h : [b, r] \rightarrow \mathbb{R}$ be defined as $h(x) = -\pi(\ell^*)(r - x) + \text{Val}_{\Gamma_{[r,e]}}(\ell^*, r)$, we set:

$$\Gamma'_{[b,r]} = \left(\text{CostConsistent} \left(\Gamma_{[b,r]}, \text{Val}_{\Gamma_{[r,e]}} \right) \right) \left[\mathbf{L}^F \cup \ell^*, h \right],$$

and compute $\text{Val}_{\Gamma'_{[b,r]}} = \text{SolveRP} \left(\Gamma'_{[b,r]} \right)$. We set:

$$f = \langle f_1, \dots, f_k \rangle = \min \{ \text{Val}_{\Gamma'_{[b,r]}}(\ell) : (\ell^*, \ell) \in \mathbf{A} \}.$$

2. Let i be the smallest natural number such that for all $j > i$ we have $f(x) \geq h(x)$ over $[b_j^f, e_j^f]$. If $i > 0$ let x^* denote the solution of $f(x) = h(x)$ (over $[b_i^f, e_i^f]$). We then set:

$$\Gamma''_{[b_i^f, x^*]} = \text{CostConsistent}(\Gamma'_{[b_i^f, x^*]}, \text{Val}_{\Gamma'_{[x^*, r]}}) \left[\mathbf{L}^F \cup \ell^*, f_i \right],$$

$$\text{and compute } \text{Val}_{\Gamma''_{[b_i^f, x^*]}} = \text{SolveRP} \left(\Gamma''_{[b_i^f, x^*]} \right).$$

3. We set:

$$\text{Val}_{\Gamma_{[b_i^f, e]}} = \text{Val}_{\Gamma''_{[b_i^f, x^*]}} \triangleright \text{Val}_{\Gamma'_{[x^*, r]}} \triangleright \text{Val}_{\Gamma_{[r, e]}}.$$

If $i = 0$ then the Γ'' term is omitted.

We set $r = b_i$. If $r \neq b$ then goto 1, otherwise output Val_{Γ_I} .

We initialise the procedure by solving the game $\Gamma_{[e, e]}$, and setting $r = e$. Observe that this can be done in polynomial time.

The following example should provide the intuition behind the iterative procedure employed during each recursive call of the algorithm.

Example 3.24. *Figure 3.3 shows how the iterative procedure in the second case of the algorithm works to compute Val_{Γ} over the interval $I = [b, e]$. In diagram a) we can see that Val_{Γ} has been computed over the interval $[r, e]$. The function h (dashed and bold line) denotes the cost function assigned to ℓ^* in Γ'' , with the affine segment f_i (dotted line), identified in Step 2 of the third case of the algorithm, just below it. The fine solid line denotes the affine segments of $f = \text{Val}_{\Gamma'}$, to the right of f_i , as computed in Step 1 of the second case of the algorithm. One can see that the affine segment f_i is such that: over the interval $[e_i, r]$ the intuition, which*

indicates that player Max should spend no time in ℓ^* , is correct; and that over the interval $[b_i, e_i]$, this intuition is not correct, i.e., it is beneficial for player Max to spend as much time as possible in ℓ^* . In diagram b) we can see the next iteration of the algorithm. Val_Γ has been computed over $[r' = b_i, e]$; this iteration follows the same steps as the previous one. Note that, over the interval $[b_i, r]$, $\text{Val}_\Gamma(\ell^*)$ is equal to f , over the interval $[e_i, r]$ and to h , over the interval $[b_i, e_i]$, as defined in Step 3 of the third case of the algorithm.

Figure 3.4 shows how the iterative procedure in the third case of the algorithm works to compute Val_Γ over the interval $I = [b, e]$. In diagram a) we can see that Val_Γ has been computed over the interval $[r, e]$. The function h denotes the cost function assigned to ℓ^* in Γ' and the dashed line denotes the function f , as defined in Step 1 of the third case of the algorithm. One can see that the interval $[b_i, e_i]$ and x^* , identified in Step 2 of the third case of the algorithm, are such that: over the interval $[x^*, r]$ the intuition, which indicates that player Min should spend all the available time in ℓ^* , is correct; and that over the interval $[b_i, x^*]$, this intuition is not correct, i.e., it is beneficial for player Min to leave ℓ^* immediately — the dashed and bold segment of f_i denotes the cost function assigned to ℓ^* in Γ'' . In diagram b) we can see the next iteration of the algorithm. Val_Γ has been computed over $[r' = b_i, e]$; this iteration follows the same steps as the previous one. Note that, over the interval $[b_i, r]$, $\text{Val}_\Gamma(\ell^*)$ is equal to h , over the interval $[x^*, r]$ and to f_i , over the interval $[b_i, x^*]$, as defined in Step 3 of the third case of the algorithm. \square

Discussion. We now briefly compare the algorithm presented in this chapter with that of Bouyer et al. [BLMR06]. The 3EXPTIME algorithm, introduced by Bouyer et al., differs from the one presented in this chapter in the way the control locations of player Min are handled. As was explained above, the algorithm presented in this chapter uses an iterative procedure, similar in spirit to that used for handling control locations of player Max. The triply-exponential algorithm, on the other

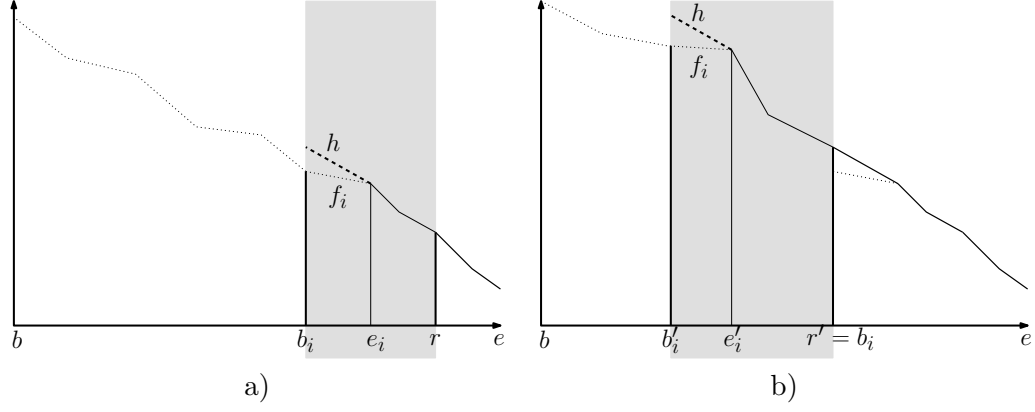


Figure 3.3: Iterative computation of $\text{Val}_\Gamma(\ell^*)$ over the interval $I = [b, e]$, where $\ell^* \in \mathcal{L}^{\text{Max}}$. Diagrams a) and b) depict two subsequent iterations; the grey rectangle indicates the subinterval for which the $\text{Val}_\Gamma(\ell^*)$ function is being computed during the given iteration.

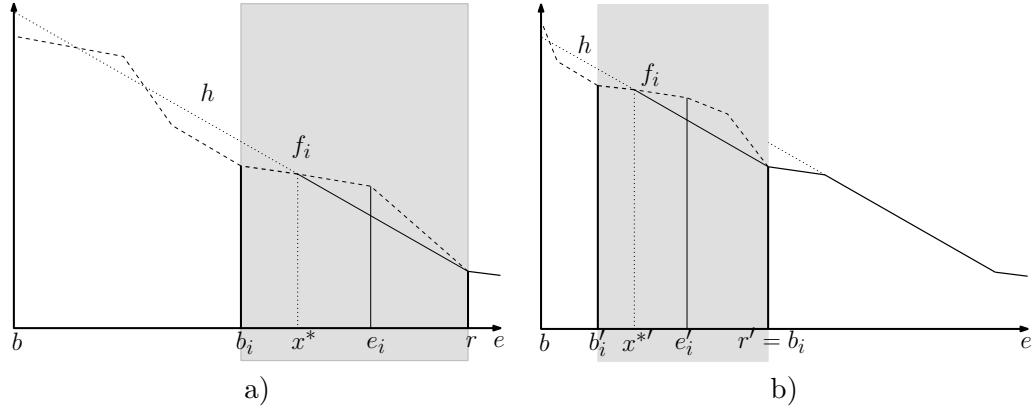


Figure 3.4: Iterative computation of $\text{Val}_\Gamma(\ell^*)$ over the interval $I = [b, e]$, where $\ell^* \in \mathcal{L}^{\text{Min}}$. Diagrams a) and b) depict two subsequent iterations; the grey rectangle indicates the subinterval for which the $\text{Val}_\Gamma(\ell^*)$ function is being computed during the given iteration.

hand, exploited the following observation: if the control location ℓ which minimises the value of the price function is visited several times, then the player Min would not be worse off if, upon the first visit, she had waited the whole time that passes between the first and last visit — this is valid because all other control locations have a higher value of the price function. This intuition is formally captured by the algorithm in the following way. Two copies of the original automaton are created, with the only difference that in both of them ℓ becomes a goal location, and hence both automata have one less non-urgent control location — this duplication introduces an exponential blowup in complexity. The first automaton captures the behaviour before ℓ is entered for the first time, whereas the second copy captures the behaviour afterwards. In the second copy ℓ is transformed into a goal location with a cost function equivalent to positive infinity — this captures the intuition that it is sufficient to visit ℓ only once. The algorithm first computes Val for the second copy. This result is used to compute $\text{Val}(\ell)$ in the first copy (a game with a single non-urgent control location). Finally, the algorithm computes Val for the first copy, with $\text{Val}(\ell)$, computed in the previous step, being assigned as a cost function to ℓ . The sought solution is Val computed for the first copy. The second exponential blowup originated from the construction that allowed to assume that the clock value is bound by 1. The construction used by Bouyer et al. [BLMR06] used control locations to encode the integer part of the clock value, and the clock itself captured only the fractional part of the clock value — this yielded an exponential blowup, as there had to be a copy of the original control location for every integer value between 0 and the value of the largest constant provided in the definition of the automaton (recall, that constants are encoded in binary). Our construction, presented in Section 3.1.1, avoids this blowup.

Computing almost optimal strategies

In the previous section we have presented an algorithm that, for a reachability-price game on a single-clock timed automaton, computes its game value, i.e., a function from states to real numbers. Although real-valued functions are not necessarily finitely representable, the function computed by the algorithm is a cost function locationwise; cost functions can be represented finitely — they are definable (recall Remark 3.13). In this section we will prove existence of ε -optimal strategies for both players, and explain how to define them.

The main result of this section is as follows.

Theorem 3.25. *Given a determined reachability-price game Γ on a single-clock timed automaton, if $\text{Val}_\Gamma(\ell)$ is a cost function for every control location ℓ , then for every $\varepsilon > 0$, positional ε -optimal strategies exist for both players.*

Proof. The theorem follows from Lemmas 3.27–3.28, which establish the existence of positional ε -optimal strategies for players Min and Max, respectively. \square

Before we state and prove Lemmas 3.27–3.28, we state the following result, which captures an essential property of the Val function.

Lemma 3.26. *Given a determined reachability-price game Γ and a control location $\ell \in \mathbb{L}^{\text{Min}}$, let:*

$$h(x) = \min\{\text{Val}(\ell', x) : (\ell, \ell') \in \mathbf{A}\},$$

we then have:

$$\text{Val}(\ell) = \min\text{C}(h, \pi(\ell)).$$

If $\ell \in \mathbb{L}^{\text{Max}}$ then, if we substitute \max for \min and $\max\text{C}$ for $\min\text{C}$, the same equality holds.

Proof. Note that determinacy of Γ guarantees that both the lower and the upper values of the game are finite. The lemma could have been strengthened by weak-

ening the assumptions to require only that the upper and lower values are finite. Determinacy would then have been a consequence of this proof.

Let Γ be a determined reachability-price game, restricted to the interval I . We prove the lemma in two steps. First, we will prove that $\text{Val}_*(\ell, x) \geq h(x) - \varepsilon$, for all $x \in I$. Second, we prove that $\text{Val}^*(\ell, x) \leq h(x) + \varepsilon$, for all $x \in I$.

The first step is to prove that $\text{Val}_*(\ell, x) \geq h(x) - \varepsilon$, for $x \in I$. Fix $\varepsilon > 0$, an ε -optimal strategy of player Max χ_ε , and a state $s = (\ell, x) \in \mathbf{S}^{\text{Min}}$. For every $\mu \in \Sigma^{\text{Min}}$, let $\omega = s \xrightarrow{t} s' \xrightarrow{(\ell, \ell')} \omega'$ denote the unique run $\text{Run}(s, \mu, \chi_\varepsilon)$ (without loss of generality we can assume that there is a single continuous transition). We then have:

$$\begin{aligned}
\mathcal{P}(\omega) &= t \cdot \pi(\ell) + \mathcal{P}(\omega') \\
&\geq t \cdot \pi(\ell) + \text{Val}((\ell', x + t)) - \varepsilon \\
&\geq t \cdot \pi(\ell) + \min_{(\ell, \ell') \in \mathbf{A}} \text{Val}((\ell', x + t)) - \varepsilon \\
&\geq \min_{0 \leq t \leq e-x} \left(t \cdot \pi(\ell) + \min_{(\ell, \ell') \in \mathbf{A}} \text{Val}((\ell', x + t)) \right) - \varepsilon \\
&= \min C(h, \pi(\ell))(x) - \varepsilon,
\end{aligned}$$

where the first inequality follows from the fact that χ_ε is ε -optimal. Due to the arbitrary choice of μ , we have obtained $\text{Val}_*(s) \geq \text{Val}_{\chi_\varepsilon}(s) \geq h(x) - \varepsilon$.

The second step of the proof, is to prove that $h(x) \geq \text{Val}^*(\ell, x) - \varepsilon$.

Fix $\varepsilon > 0$, and μ_ε , an ε -optimal strategy of player Min,. We define μ^* in the following way:

$$\mu^*(s') = \begin{cases} t^* & \text{if } s = (\ell, x), \\ \ell^* & \text{if } s = (\ell, x + t^*), \\ \mu_\varepsilon(s') & \text{otherwise,} \end{cases}$$

where:

$$\begin{aligned} t^* &= \arg \min_{0 \leq t \leq e-x} t \cdot \pi(\ell) + h(x+t), \\ \ell^* &= \arg \min_{(\ell, \ell') \in \mathbf{A}} \text{Val}(\ell', x+t^*). \end{aligned}$$

To finish the proof, we now show that μ^* is a witness to the $h(x) \geq \text{Val}(\ell, x) - \varepsilon$ inequality. Consider a run $\omega = s \xrightarrow{t^*} s' \xrightarrow{(\ell, \ell')} \omega' = \text{Run}(s, \mu^*, \chi)$, for some $\chi \in \Sigma^{\text{Max}}$. We then have:

$$\begin{aligned} \mathcal{P}(\omega) &= t^* \cdot \pi(\ell) + \mathcal{P}(\omega') \\ &\leq t^* \cdot \pi(\ell) + \text{Val}(\ell^*, x+t^*) + \varepsilon \\ &= \min C(h, \pi(\ell)) + \varepsilon. \end{aligned}$$

The inequality is a consequence of the fact that μ^* is an ε -optimal strategy in a reachability-price game starting in state $(\ell^*, x+t^*)$. The final equality follows from the definition of the h function and the $\min C$ operator. Due to the arbitrary choice of χ , we have established $h(x) + \varepsilon \geq \text{Val}^{\mu^*}(\ell, x) \geq \text{Val}^*(\ell, x)$.

We have obtained the desired result. Analogous reasoning allows to establish equality for the control locations of player Max. \square

Lemma 3.27. *Provided that Val is a cost function locationwise, there exists a natural number K such that for every $\varepsilon > 0$, there exists a positional ε -optimal strategy $\mu_\varepsilon \in \Sigma^{\text{Min}}$ such that the number of discrete transitions in the run $\omega_{\text{Stop}(\omega)}$ is not greater than K , where $\omega = \text{Run}(s, \mu_\varepsilon, \chi)$, for every $s \in S$ and for every $\chi \in \Sigma^{\text{Max}}$.*

Lemma 3.28. *Provided that Val is a cost function locationwise, for every $\varepsilon > 0$, there exists a positional ε -optimal strategy $\chi_\varepsilon \in \Sigma^{\text{Max}}$.*

We omit the proof of Lemma 3.28, as it is similar to the proof of Lemma 3.27,

with the only difference that we do not need to guarantee a bound on the number of discrete transitions.

Proof of Lemma 3.27. To construct an ε -optimal positional strategy, we are relying on the fact that the game is determined and that the value function is a cost function locationwise.

We will say, that a strategy $\mu \in \Sigma^{\text{Min}}$ has a step-bound K in state s , if for every strategy $\chi \in \Sigma^{\text{Max}}$ and every run $\omega = \text{Run}(s, \mu, \chi)$, the number of discrete transitions in the run $\omega_{\text{Stop}(\omega)}$ is less or equal to K . We say that a strategy has a step-bound K , if it has a step-bound K in every state.

To prove the lemma, we will construct a positional strategy $\mu^* \in \Sigma^{\text{Min}}$, and prove that it is ε -optimal for some $\varepsilon > 0$, and that it has a step-bound. We assume that Γ is determined, and that $\text{Val}(\ell)$ is a cost function for every control location ℓ . This allows us to partition the interval $I = [b, e]$, over which the game is played, into a finite number of intervals I_1, \dots, I_k in such a way that for every $i = 1, \dots, k$, and for every $\ell \in \mathbf{L}$, the function $\text{Val}(\ell)|_{I_i}$ is affine.

We start with the following observation. In runs from states having e as the clock value only discrete transitions are admissible, i.e., we are dealing with a game on a finite graph. Clearly (see Section 2.2.2), there exist a natural number K and a positional ε -optimal strategy $\mu_\varepsilon \in \Sigma^{\text{Min}}$ for every $\varepsilon > 0$, that has a bound K in all states with clock value e .

We will now define a function $D : \mathbf{S} \rightarrow \mathbb{N}$, which will aid our definition of the strategy μ^* . Intuitively, the function D measures the “distance” to a goal state, where the distance measures the necessary number of discrete transitions in a run induced by an ε -optimal strategy. The key concept behind the definition, formally captured by Lemma 3.26, is as follows. If the slope of the value function in the given control location and at the given clock value is equal to the negative of the price of that control location, then it is beneficial for the player to wait (as long as the slope remains equal) — the distance does not increase. Only when the slope is different,

it is beneficial for a player to perform a discrete transition, to a state with the same game value, but with a smaller distance.

We define a function $D : \mathbf{S} \rightarrow \mathbb{N}$ in the following way. For $\ell \in \mathbf{L}^F$, we set $D(\ell, \cdot) = 0$; for $\ell \in \mathbf{L}^{\text{Min}} \setminus \mathbf{L}^F$, we set:

$$D((\ell, x)) = \begin{cases} 0 & \text{if } x = e, \\ 1 + \min \left\{ D(\ell', x) : (\ell, \ell') \in \mathbf{A} \text{ and } \right. & \text{if } t_{(\ell, x)}^* = 0, \\ \quad \left. \text{Val}(\ell, x) = \text{Val}(\ell', x) \right\} & \\ D(\ell, x + t_{(\ell, x)}^*) & \text{if } t_{(\ell, x)}^* \neq 0; \end{cases}$$

and for $\ell \in \mathbf{L}^{\text{Max}} \setminus \mathbf{L}^F$, we set:

$$D((\ell, x)) = \begin{cases} 0 & \text{if } x = e, \\ \max \left\{ 1 + \max \left\{ D(\ell', x) : \right. \right. & \text{otherwise,} \\ \quad \left. \left. (\ell, \ell') \in \mathbf{A} \text{ and } \text{Val}(\ell, x) = \text{Val}(\ell', x) \right\}, \right. & \\ \quad \left. D(\ell, x + t_{(\ell, x)}^*) \right\} & \end{cases}$$

where

$$t_{(\ell, x)}^* = \max \left\{ t : \text{Val}(\ell, x + t) = \text{Val}(\ell, x) - \pi(\ell) \cdot t \text{ and } t \in \mathbf{T} \right\}.$$

The definition of the $t_{(\ell, x)}^*$ value is similar to that of t^* , found in the proof of Lemma 3.26, and characterises the “optimal” time to spend in control location ℓ , given that the clock value is x . The definition of the D function is different depending on the ownership of the control location. In the case of player Min, we assume that she wants to minimise the number of discrete transitions; she chooses such transitions only when it is strictly necessary. In the case of player Max, we assume the opposite; he chooses a discrete transition whenever possible, and in such a way, as to maximise the distance to the goal location — this is consistent with his

game goal.

We now explain why the D function is bounded. First, observe that its value is finite for every state s (by Assumption 3.12). Second, observe that, for every control location ℓ , the value of the function $D((\ell, \cdot))$ is constant over $[b_i, e_i)$, for every $i \in \{1, \dots, k\}$.

We now define the positional strategy μ^* , as postulated in the lemma statement.

$$\mu^*(\ell, x) = \begin{cases} \mu_\varepsilon(\ell, x) & \text{if } x = e, \\ \left(\ell, \ell_{(\ell, x)}^*\right) & \text{if } t_{(\ell, x)}^* = 0, \\ t_{(\ell, x)}^* & \text{otherwise,} \end{cases}$$

where μ_ε is the positional ε -optimal strategy that has bound K in states with clock value e ; $t_{(\ell, x)}^*$ is defined as before; and the value of $\ell_{(\ell, x)}^*$ is defined as:

$$\begin{aligned} \ell_{(\ell, x)}^* &= \arg \min_{\ell'} \left\{ D(\ell', I_i) : \right. \\ &\quad \left. (\ell, \ell') \in \mathbf{A}, \text{Val}(\ell, x) = \text{Val}(\ell', x), x \in I_i, \text{ and } i \in \{1, \dots, k\} \right\}. \end{aligned}$$

Notice that μ^* is guided by the distance function. First, it prescribes a discrete transition only if it is strictly necessary. Second, the discrete transition prescribed is such that the state reached has the smallest distance possible.

Before we start proving that μ^* is ε -optimal, and that it has a finite bound, we discuss its main properties. Recall that Lemma 3.26, together with Proposition 3.18, establishes that for $\ell \in \mathbf{L}^{\text{Min}}$, the slope of the affine segments of $\text{Val}(\ell)$ are not steeper than $-\pi(\ell)$. This allows us to establish that for every clock value $x \in I$, if we set $t = t_{(\ell, x)}^*$ then the value of the following expression is minimised:

$$\text{Val}(\ell, x + t) + \pi(\ell) \cdot t,$$

and that for $t > t_{(\ell, x)}^*$ the value of this expression is strictly greater. This allows us

establish that for every state $s \in S^{\text{Min}}$ and a transition $s \xrightarrow{\mu^*(s)} s'$, if $\mu^*(s) \in \mathsf{T}$ then we have:

$$\begin{aligned}\text{Val}(s) &= t_s^* \cdot \pi(\ell) + \text{Val}(s'), \\ D(s) &= D(s'),\end{aligned}$$

or, if $\mu^*(s) \in \mathsf{A}$ then we have:

$$\begin{aligned}\text{Val}(s) &= \text{Val}(s'), \\ D(s) &= 1 + D(s').\end{aligned}$$

We now proceed to prove that μ^* is ε -optimal, and that it has a step-bound of $N = \sup_{s \in S} D(s) + K$. Consider a run $\omega = \text{Run}(s, \mu^*, \chi)$, starting in some state s , induced by μ^* and some strategy $\chi \in \Sigma^{\text{Max}}$. For every transition $s_i \xrightarrow{\lambda_{i+1}} s_{i+1}$ we have that if $\lambda_{i+1} \in \mathsf{T}$ then

$$\begin{aligned}\text{Val}(s_i) &\geq \lambda_{i+1} \cdot \pi(\ell) + \text{Val}(s_i), \\ D(s_i) &\geq D(s_i),\end{aligned}$$

or, if $\lambda_{i+1} \in \mathsf{A}$, then:

$$\begin{aligned}\text{Val}(s_i) &\geq \text{Val}(s_{i+1}), \\ D(s_i) &\geq 1 + D(s_{i+1}).\end{aligned}$$

Let $n = \min\{i : s_i = (\ell, e), \text{ for some } \ell \in \mathsf{L}\}$. Notice that $n \leq D(s)$. If we sum up the n inequalities, we obtain

$$\text{Val}(s) \geq \text{Price}(\omega_n) + \text{Val}(s_n).$$

Using the fact that the clock value in s_n is equal to e_k , and that μ_ε is ε -optimal, we obtain:

$$\text{Val}(s) \geq \text{Price}(\omega_n) + \mathcal{P}(\text{Run}(\omega_n, \mu^*, \chi)) - \varepsilon = \mathcal{P}(\omega) - \varepsilon.$$

Notice that ω has at most $D(s) + K$ discrete transitions before a goal state is reached. This, due to the arbitrary choice of χ , the strategy of player Max, yields the desired result, i.e., that $\text{Val}(s) \geq \text{Val}^{\mu^*}(s) - \varepsilon$, and that μ^* has a step-bound $N = \sup_{s \in S} D(s) + K$. (recall that the function D is bounded). \square

Corollary 3.29. *Positional ε -optimal strategies for both players are definable.*

Proof. The Corollary follows from the proofs of Lemmas 3.27–3.28. \square

3.3.2 Correctness and complexity

In this section we will provide a proof that the algorithm (introduced in Section 3.3.1) for computing the value of reachability-price games on single-clock timed automata is correct, and as a consequence, that these games are positionally determined. Determinacy of reachability-price games will imply the existence and computability of almost optimal positional strategies for both players (a consequence of the results introduced in Section 3.3.1).

The main results presented in this section are that the procedure **SolveRP**, introduced in Section 3.3.1, correctly computes the value of a reachability-price game on a single-clock timed automaton (Theorem 3.32), and that the running time of the algorithm is at most exponential (Theorem 3.38).

The section is organised as follows. First we establish correctness of some auxiliary operations used by the algorithm. Second, we state and prove Theorem 3.32, i.e., that the algorithm correctly computes game values. Third, we conclude this section with the statement and proof of Theorem 3.38, i.e., that the algorithm runs in exponential time.

We start by proving the correctness of the **CostConsistent** operation, introduced in Section 3.3.1. Fix a game Γ and two intervals $I_1 = [b_1, e_1]$ and $I_2 = [b_2, e_2]$ such that $r = e_1 = b_2$. Recall that the purpose of the **CostConsistent** operation is to enable us compute $\text{Val}_{\Gamma_{I_1 \cup I_2}}$, provided that we have already computed $\text{Val}_{\Gamma_{I_2}}$. The intuitive correctness of the **CostConsistent** operation is formalised by the following lemma.

Lemma 3.30. *If $\Gamma'_{I_1} = \text{CostConsistent}(\Gamma_{I_1}, \text{Val}_{\Gamma_{I_2}})$ then, for every $\ell \in \mathbf{L}$,*

$$\text{Val}_{\Gamma'_{I_1}}(\ell) \triangleright \text{Val}_{\Gamma_{I_2}}(\ell) = \text{Val}_{\Gamma_{I_1 \cup I_2}}(\ell),$$

provided that Γ_{I_1} and Γ_{I_2} are determined.

Proof. To prove the lemma, we will first prove that:

$$\text{Val}_{\Gamma'_{I_1}}(\ell)(r) = \text{Val}_{\Gamma_{I_2}}(\ell)(r)$$

and then proceed to prove that:

$$\text{Val}_{\Gamma'_{I_1}}(\ell) \triangleright \text{Val}_{\Gamma_{I_2}}(\ell) = \text{Val}_{\Gamma_{I_1 \cup I_2}}(\ell),$$

for $\ell \in \mathbf{L}$.

We start by proving that:

$$\text{Val}_{\Gamma'_{I_1}}(\ell)(r) \leq \text{Val}_{\Gamma_{I_2}}(\ell)(r),$$

for all $\ell \in \mathbf{L}$. Clearly, the inequality holds for all $\ell \in \mathbf{L}^{\text{Min}}$ because in Γ' we have the transition (ℓ, ℓ') and $\text{CF}^{\text{F}'}(\ell')(r) = \text{Val}_{\Gamma_{I_2}}(\ell)(r)$. It now remains to argue that the same inequality holds for $\ell \in \mathbf{L}^{\text{Max}}$. First, observe that if for all states (ℓ'', r) such that $(\ell, \ell'') \in \mathbf{A}$, we have $\text{Val}_{\Gamma'_{I_1}}(\ell'')(r) \leq \text{Val}_{\Gamma_{I_2}}(\ell'')(r)$, then $\text{Val}_{\Gamma'_{I_1}}(\ell)(r) \leq \text{Val}_{\Gamma_{I_2}}(\ell)(r)$. Second, recall that we are considering games where player Min can

ensure that a goal location can be reached from every control location. These two observations allow us to iteratively show the sought upper bound for all control locations $\ell \in \mathcal{L}^{\text{Max}}$. We start by showing the bound for control locations whose successors are either goal or player Min's control locations. In subsequent iterations we show the bound for control locations whose successors are either goal, player Min's, or processed player Max's control locations. We use the first observation to show the upper bound. The second observation allows us to establish that in each iteration we process, i.e., show the bound, at least one new control location. If we could not do this, this would mean that player Max can enforce a cycle on non-goal control locations, which in turn would contradict the assumption that player Min can enforce reaching a goal location. This way we have shown that the inequality holds in all control locations.

To prove the opposite inequality, i.e.,

$$\text{Val}_{\Gamma'_{I_1}}(\ell)(r) \geq \text{Val}_{\Gamma_{I_2}}(\ell)(r),$$

we proceed in a similar fashion as above. However, we need to observe that, unlike player Max, player Min can potentially form “cycles of control locations”. To alleviate this problem, we point out that in a situation where time can not progress, a cycle is not beneficial for player Min. This allows us to apply the iterative proof technique used in the proof of the \leq inequality.

From the fact that Γ is simple we have

$$\text{Val}_{\Gamma_{I_2}} = \left(\text{Val}_{\Gamma_{I_1 \cup I_2}} \right)_{|I_2}.$$

It remains to show that

$$\text{Val}_{\Gamma'_{I_1}} = \left(\text{Val}_{\Gamma_{I_1 \cup I_2}} \right)_{|I_1}.$$

We prove this by showing that if in one game a player can choose a strategy that

assures a certain payoff, then by switching to the other game the payoff would not improve. We show this through an analysis of game runs. It suffices to consider runs that visit a state in $S_{I_1} \cap S_{I_2}$.

Consider a run ω in the game $\Gamma_{I_1 \cup I_2}$, and a state s_i along that run, such that $s_i \in S_{I_1}$ and $s_{i+1} \in S_{I_2}$. We can transform ω into ω' by introducing an additional state $s'_i \in S_{I_1} \cap S_{I_2}$ and substituting the transition $s_i \xrightarrow{t_{i+1}} s_{i+1}$ by $s_i \xrightarrow{t'} s'_i \xrightarrow{t''} s_{i+1}$. Since $t_{i+1} = t' + t''$, the price, visited control locations and the final state of ω' are the same as of ω . From this we can conclude that in order to establish $\text{Val}_{\Gamma_{I_1 \cup I_2}}$, it is without loss of generality if we only consider strategies with the following property. For every state $s \in S_{I_1}$, if a run generated by these strategies visits a state $s' \in S_{I_2}$, then it visits a state in $S_{I_1} \cap S_{I_2}$.

Fix $\varepsilon > 0$. Take strategies $\mu_\varepsilon \in \Sigma^{\text{Min}}$ and $\chi_\varepsilon \in \Sigma^{\text{Max}}$ that are ε -optimal in $\Gamma_{I_1 \cup I_2}$ for players Min and Max respectively. Given a state s , consider $\omega = \text{Run}(s, \mu_\varepsilon, \chi_\varepsilon)$. As outlined earlier, we can view ω as a concatenation of two runs ω_1 with all states in S_{I_1} and ω_2 with all states in S_{I_2} , with the ending state of ω_1 (the initial state of ω_2) in $S_{I_1} \cap S_{I_2}$ — let us denote this state by s_ω . Thus, for every state $s \in S_{I_1}$, we have:

$$\begin{aligned}
\text{Val}_{\Gamma_{I_1 \cup I_2}}(s) &\geq \mathcal{P}_{\Gamma_{I_1 \cup I_2}}(\omega) - \varepsilon \\
&\geq \text{Price}(\omega_1) + \mathcal{P}_{\Gamma_{I_1 \cup I_2}}(\omega_2) - \varepsilon \\
&\geq \text{Price}(\omega_1) + \text{Val}_{\Gamma_{I_1 \cup I_2}}(s_\omega) - 2\varepsilon \\
&= \text{Price}(\omega_1) + \text{Val}_{\Gamma'_{I_1}}(s_\omega) - 2 \cdot \varepsilon \\
&= \mathcal{P}_{\Gamma'_{I_1}}(\omega_1) - 2 \cdot \varepsilon \\
&\geq \text{Val}_{\Gamma'_{I_1}}(s) - 3 \cdot \varepsilon,
\end{aligned}$$

We have shown that $(\text{Val}_{\Gamma_{I_1 \cup I_2}})_{|I_1}(s) \geq \text{Val}_{\Gamma'_{I_1}}(s)$, for every state $s \in S$. To finish the proof, we need to show that $(\text{Val}_{\Gamma_{I_1 \cup I_2}})_{|I_1}(s) \leq \text{Val}_{\Gamma'_{I_1}}(s)$, for every state $s \in S$. We prove this in a similar fashion, by showing that $\text{Val}_{\Gamma_{I_1 \cup I_2}}(s) \leq \text{Val}_{\Gamma'_{I_1}}(s) + 3 \cdot \varepsilon$,

for every state $s \in S$. □

The second operation used by the **SolveRP** procedure is an operation that allows to compute the **Val** function for some determined game Γ , provided that for some control location ℓ , the function has already been computed, i.e., that the function $x \mapsto \text{Val}(\ell, x)$ is known. We now formally establish correctness of this operation.

Lemma 3.31. *Given a determined reachability-price game Γ over an interval I , a control location ℓ , and a cost function $h : I \rightarrow \mathbb{R}_{\geq 0}$, if $h(x) = \text{Val}_\Gamma(\ell, x)$, for every $x \in I$, then:*

$$\text{Val}_{\Gamma[\mathbb{L}^F \cup \ell, h]}(\ell', x) = \text{Val}_\Gamma(\ell', x),$$

for every control location $\ell' \in \mathbb{L}$ and every clock valuation $x \in I$.

Proof. Lemma 3.31 is a direct consequence of Lemma 3.26, which characterises the relation between the game values in adjacent control locations. □

We show that the procedure **SolveRP** is correct, i.e., that if it terminates, the output is in fact the **Val** function, and that it indeed terminates. Later, we will also show that there is an exponential upper bound on the running time of **SolveRP**.

Theorem 3.32. *Given a reachability-price game Γ , the **SolveRP**(Γ) procedure terminates and outputs the function Val_Γ , which is a cost function at every control location.*

We will prove the theorem in two steps. First, we prove that if the iterative procedure in cases 2 and 3 terminates, it computes **Val** (Theorem 3.33). Second, we show that it always terminates (Theorem 3.37), i.e., we first show the partial, and then the full correctness of the algorithm. Notice that this algorithmic result implies determinacy of Γ .

Theorem 3.33. *Given a reachability-price game Γ , if **SolveRP**(Γ) procedure terminates, it outputs the function Val_Γ , which is a cost function at every control location.*

Proof. The proof is inductive. Fix Γ and let ℓ be the non-urgent control location that minimises $\pi(\ell)$. Assume that Γ has $n + 1$ non-urgent control locations and that Theorem 3.33 holds for every game Γ' that has at most n non-urgent control locations.

If there are only urgent control locations then computing Val_Γ amounts to solving a reachability-price game on a finite graph. It remains to prove the inductive step. There are two cases to consider: the first case when $\ell \in \mathbf{L}^{\text{Min}}$ and the second case when $\ell \in \mathbf{L}^{\text{Max}}$. The proof of these two cases follows from Lemmas 3.34 and 3.36. \square

First we consider the case when $\ell \in \mathbf{L}^{\text{Max}}$. This case is handled in the same way as in the algorithm of Bouyer et al. [BLMR06].

Lemma 3.34. *Given a reachability-price game in which the control location with the smallest value of the price function is in \mathbf{L}^{Max} , if the second case of **SolveRP** procedure terminates, then it outputs Val_Γ , which is a cost function at every control location.*

Before we begin the actual proof of Lemma 3.34, we state the following instrumental fact.

Lemma 3.35. *If the reachability-price game Γ is determined, then for every $\varepsilon > 0$ and for every ε -optimal strategy of player Min, we have that for every $\chi \in \Sigma^{\text{Max}}$ and every $s \in \mathbf{S}$ the following holds:*

$$\text{Val}(s) + \varepsilon \geq \text{Price}(\omega_k) + \text{Val}(s_k),$$

where $\omega = \text{Run}(s, \mu_\varepsilon, \chi)$ visits a goal state, and $k \leq \min\{\text{Len}(\omega), \text{Stop}(\omega)\}$.

Proof. The proof is straightforward. Take $\varepsilon > 0$, and an ε -optimal strategy of player

Min μ_ε . Let $\omega = \text{Run}(s, \mu_\varepsilon, \chi)$ for some $\chi \in \Sigma^{\text{Max}}$ and some state s . We have:

$$\text{Val}(s) + \varepsilon \geq \text{Val}^{\mu_\varepsilon}(s) \geq \mathcal{P}(\omega).$$

For every $k \leq \min\{\text{Len}(\omega), \text{Stop}(\omega)\}$, we have:

$$\text{Val}(s) + \varepsilon \geq \mathcal{P}(\omega) = \text{Price}(\omega_k) + \mathcal{P}(\text{Run}(\omega_k, \mu_\varepsilon, \chi)).$$

The choice of χ was arbitrary, hence if we take supremum over χ we obtain

$$\text{Val}(s) + \varepsilon \geq \text{Price}(\omega_k) + \text{Val}^{\mu_\varepsilon}(s) \geq \text{Price}(\omega_k) + \text{Val}(s),$$

which, due to the arbitrary choice of s , yields the desired result. \square

Now we can complete the proof of Lemma 3.34.

Proof of Lemma 3.34. We will show that the single iteration of the second case of the algorithm correctly computes Val_Γ over the given interval. Without loss of generality we assume that we are dealing with a single interval $I = [b, e]$ (in the actual algorithm, it would have been $I_i = [b_i, e_i = r]$). Lemma 3.30 allows us to establish that results computed in subsequent operations are combined correctly.

Let ℓ^* , Γ' , Γ'' , be defined as in the second case of the algorithm. There are two cases to consider: the first case, in which the slope of $\text{Val}_{\Gamma'}(\ell^*)$ is steeper than or equal to $-\pi(l)$, and the second case, in which the slope of $\text{Val}_{\Gamma'}(\ell^*)$ is shallower than $-\pi(\ell^*)$. In the first case we will show that $\text{Val}_{\Gamma'}(s) = \text{Val}_\Gamma(s)$ for every state s , and in the second case we will show that $\text{Val}_{\Gamma''}(s) = \text{Val}_\Gamma(s)$ for every state s . To achieve this goal, we will show that Val_Γ exists, i.e., that Γ is determined, and that that it is indeed equal to $\text{Val}_{\Gamma'}$ and $\text{Val}_{\Gamma''}$, respectively. In particular, this implies that Val_Γ is a cost function at every control location. In the proof we will be relying on the inductive hypothesis that the games with one less non-urgent control location are

determined, that the Val function for such games is a cost function at every control location, and that the algorithm computes the game value correctly for such games. Note that Γ' and Γ'' have one less non-urgent control location than Γ .

The games Γ' and Γ'' can be simulated in Γ by choosing an appropriate strategy $\chi \in \Sigma^{\text{Max}}$. In the first case this strategy would leave ℓ^* immediately, and in the second case this strategy would spend all available time in ℓ^* . We use this observation to establish that, depending on the case:

$$\text{Val}_{\Gamma'}(s) \leq \text{Val}_{*\Gamma}(s) \text{ or } \text{Val}_{\Gamma''}(s) \leq \text{Val}_{*\Gamma}(s),$$

for every $s \in \mathbf{S}$. Recall the definition of Val_* , which states that for every $\chi \in \Sigma^{\text{Max}}$, and every $s \in \mathbf{S}$, we have that $\text{Val}_\chi(s) \leq \text{Val}_{*\Gamma}(s)$.

To show determinacy of Γ , and to prove that Val_Γ is indeed computed correctly, it remains to show that, depending on the case, $\text{Val}_{*\Gamma}(s) \leq \text{Val}_{\Gamma'}(s)$ or $\text{Val}_{*\Gamma}(s) \leq \text{Val}_{\Gamma''}(s)$ for all states s . To achieve this, in the case of Γ' , we choose an $\varepsilon > 0$, and we fix a strategy $\mu_\varepsilon \in \Sigma^{\text{Min}}$, which has some desired properties, and show that for every $\chi \in \Sigma^{\text{Max}}$, and every state $s \in \mathbf{S}$ we have that $\mathcal{P}_\Gamma(\text{Run}(s, \mu_\varepsilon, \chi)) \leq \text{Val}_{\Gamma'}(s) + \varepsilon$. This, due to the arbitrary choice of χ , establishes that $\text{Val}_{\Gamma}^{\mu_\varepsilon}(s) \leq \text{Val}_{\Gamma'}(s) + \varepsilon$, this in turn, by definition of Val^* , establishes that $\text{Val}_\Gamma^*(s) \leq \text{Val}_{\Gamma'}(s)$, which yields the desired result. The case of Γ'' is handled analogously.

We will restrict the set of states to $(\{\ell^*\} \times I) \cap \mathbf{S}$, and apply Lemma 3.31 to establish that the result holds for all states.

There are two cases to consider. We start with the first, i.e., the case when the slope of $\text{Val}_{\Gamma'}(\ell^*)$ is steeper than or equal than $-\pi(\ell^*)$ over I . We will show that $\text{Val}_\Gamma^*(\ell^*)(x) \leq \text{Val}_{\Gamma'}(\ell^*)(x)$, for $x \in I$.

Before we proceed with the proof, we introduce a simplifying assumption on the structure of the runs. Observe that a finite number of consecutive continu-

ous transitions in one control location can be substituted with a single continuous transition, which bears the same price as the accumulated price of the substituted transitions — this is a consequence of the fact that the price function is linear with respect to the duration of a transition. This observation, together with Remark 2.33, allows us, without the loss of generality, to restrict our consideration to runs which are alternating sequences of discrete and continuous transitions.

Let K be the upper bound on the number of visits to a control location in Γ' postulated in Lemma 3.27. Given $\varepsilon' \leq \varepsilon/K$, let $\mu_{\varepsilon'} \in \Sigma^{\text{Min}}$ be an ε' -optimal strategy, in Γ' that has a step-bound K . Notice that $\mu_{\varepsilon'}$ is a valid Γ strategy, with the same step-bound. For every $\chi \in \Sigma^{\text{Max}}$ in Γ , and for every $s \in \{\ell^*\} \times I$, let ω denote the unique run $\text{Run}(s, \mu_{\varepsilon'}, \chi)$, and let's assume it visits ℓ^* exactly m times, after transitions i_1, \dots, i_m . We then have the following:

$$\begin{aligned}
\mathcal{P}_{\Gamma}(\omega) &= \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_{\Gamma}(\text{Run}(\omega_{i_m+1}, \mu_{\varepsilon'}, \chi)) \\
&\leq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_{\Gamma'}(\text{Run}(\omega_{i_m+2}, \mu_{\varepsilon'}, \chi)) \\
&\leq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_m+2}) + \varepsilon' \\
&\leq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_m+1}) + \varepsilon' \\
&\leq \text{Price}(\omega_{i_m}) + \text{Val}_{\Gamma'}(s_{i_m}) + \varepsilon',
\end{aligned}$$

where $s_i = (\ell_i, x_i)$. The first inequality holds because λ_{i_m+2} is a discrete transition, which does not incur a price, and because the suffix of ω starting in s_{i_m+2} does not visit ℓ^* . The second inequality holds because $\mu_{\varepsilon'}$ is an ε' -optimal strategy of player Min in Γ' . The third inequality holds because, by definition (ℓ^* is an urgent control location),

$$\text{Val}_{\Gamma'}(\ell^*)(x) = \max_{(\ell^*, \ell) \in A'} \text{Val}_{\Gamma'}(\ell)(x)$$

over the interval I . The fourth inequality holds because, in the case being considered, the slope of $\text{Val}_{\Gamma'}(\ell^*)$ is steeper than or equal to $-\pi(\ell^*)$ over the interval I .

If $m > 1$ then we continue the proof in an inductive fashion. Assume that we have shown

$$\mathcal{P}_\Gamma(\omega) \leq \text{Price}(\omega_{i_{j+1}}) + \text{Val}_{\Gamma'}(s_{i_{j+1}}) + (m - j) \cdot \varepsilon',$$

for some $j \in \{1, \dots, m - 1\}$. We then have:

$$\begin{aligned} \mathcal{P}_\Gamma(\omega) &\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) \\ &\quad + \text{Price}\left(\left(\text{Run}(\omega_{i_j+1}, \mu_\varepsilon, \chi)\right)_{i_{j+1}-i_j-1}\right) \\ &\quad + \text{Val}_{\Gamma'}(s_{i_{j+1}}) + (m - i) \cdot \varepsilon' \\ &\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) \\ &\quad + \text{Price}\left(\left(\text{Run}(\omega_{i_j+2}, \mu_\varepsilon, \chi)\right)_{i_{j+1}-i_j-2}\right) \\ &\quad + \text{Val}_{\Gamma'}(s_{i_{j+1}}) + (m - i) \cdot \varepsilon' \\ &\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_{j+2}}) + (m - i + 1) \cdot \varepsilon' \\ &\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_{j+1}}) + (m - i + 1) \cdot \varepsilon' \\ &\leq \text{Price}(\omega_{i_j}) + \text{Val}_{\Gamma'}(s_{i_{j+1}}) + (m - i + 1) \cdot \varepsilon' \end{aligned}$$

Inequalities 2–5, with the exception of inequality 3, are justified in the same fashion as inequalities 1–4, in the first step of the proof of this case. We now justify inequality 3. Note that

$$\omega' = \text{Run}(\omega_{i_j+2}, \mu'_\varepsilon, \chi)_{i_{j+1}-i_j-2}$$

visits ℓ^* only upon entering $s_{i_{j+1}}$, hence $k = i_{j+1} - i_j - 2 \leq \min\{\text{Len}(\omega'), \text{Stop}_{\Gamma'}(\omega')\}$ — we can therefore apply Lemma 3.35.

To finish the proof of this case, we observe that $\text{Len}(\omega_{i_1}) = 0$ and, as a consequence, $\text{Price}_\Gamma(\omega_{i_1}) = 0$. Hence, we have shown that $\mathcal{P}_\Gamma(\omega) \leq \text{Val}_{\Gamma'}(s) + \varepsilon$.

We now proceed to the second case, i.e., the case when the slope of $\text{Val}_{\Gamma'}(\ell^*)$

is shallower than $-\pi(\ell^*)$, over I . We will show that $\text{Val}_\Gamma^*(\ell^*)(x) \leq \text{Val}_{\Gamma''}(\ell^*)(x)$, for $x \in I$ (recall that the slope of $\text{Val}_{\Gamma''}(\ell^*)$ is equal to $-\pi(\ell^*)$ over I).

Let K be the upper bound on the number of visits to a control location in Γ'' , postulated in Lemma 3.27. Given $\varepsilon' \leq \varepsilon/K$, let $\mu_{\varepsilon'} \in \Sigma^{\text{Min}}$ be ε' -optimal strategy, in Γ'' that has a step-bound K . Notice that $\mu_{\varepsilon'}$ is a valid Γ strategy with the same step-bound. For every $\chi \in \Sigma^{\text{Max}}$ in Γ , and for every $s \in \{\ell^*\} \times I$, let ω denote the unique run $\text{Run}(s, \mu_{\varepsilon'}, \chi)$, and let's assume it visits ℓ^* exactly m times, after transitions i_1, \dots, i_m . We then have the following:

$$\begin{aligned}
\mathcal{P}_\Gamma(\omega) &= \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_\Gamma(\text{Run}(\omega_{i_m+1}, \mu_{\varepsilon'}, \chi)) \\
&\leq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_{\Gamma''}(\text{Run}(\omega_{i_m+2}, \mu_{\varepsilon'}, \chi)) \\
&\leq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_m+2}) + \varepsilon' \\
&\leq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_m+1}) + \varepsilon' \\
&\leq \text{Price}(\omega_{i_m}) + \text{Val}_{\Gamma''}(s_{i_m}) + \varepsilon',
\end{aligned}$$

where $s_i = (\ell_i, x_i)$. The first inequality holds because λ_{i_m+2} is a discrete transition, which does not incur a price, and because the suffix of ω starting in s_{i_m+2} does not visit ℓ^* . The second inequality holds because $\mu_{\varepsilon'}$ is an ε' -optimal strategy of player Min in Γ'' . The third inequality holds because, by definition,

$$\text{Val}_{\Gamma''}(\ell^*)(x) \geq \max_{(\ell^*, \ell) \in \mathbf{A}'} \text{Val}_{\Gamma''}(\ell)(x)$$

over the interval I . The fourth inequality holds because, in the case being considered, the slope of $\text{Val}_{\Gamma''}(\ell^*)$ is equal to $-\pi(\ell^*)$ over the interval I .

If $m > 1$, then we continue the proof in an inductive fashion. Assume that we have shown

$$\mathcal{P}_\Gamma(\omega) \leq \text{Price}(\omega_{i_{j+1}}) + \text{Val}_{\Gamma''}(s_{i_{j+1}}) + (m - j) \cdot \varepsilon',$$

for some $j \in \{1, \dots, m-1\}$. We then have:

$$\begin{aligned}
\mathcal{P}_\Gamma(\omega) &\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) \\
&\quad + \text{Price}\left(\left(\text{Run}(\omega_{i_j+1}, \mu_\varepsilon, \chi)\right)_{i_{j+1}-i_j-1}\right) \\
&\quad + \text{Val}_{\Gamma''}(s_{i_{j+1}}) + (m-i) \cdot \varepsilon' \\
&\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) \\
&\quad + \text{Price}\left(\left(\text{Run}(\omega_{i_j+2}, \mu_\varepsilon, \chi)\right)_{i_{j+1}-i_j-2}\right) \\
&\quad + \text{Val}_{\Gamma''}(s_{i_{j+1}}) + (m-i) \cdot \varepsilon' \\
&\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_{j+2}}) + (m-i+1) \cdot \varepsilon' \\
&\leq \text{Price}(\omega_{i_j}) + \lambda_{i_j+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_{j+1}}) + (m-i+1) \cdot \varepsilon' \\
&\leq \text{Price}(\omega_{i_j}) + \text{Val}_{\Gamma'}(s_{i_{j+1}}) + (m-i+1) \cdot \varepsilon'
\end{aligned}$$

Inequalities 2–5, with the exception of inequality 3, are justified in the same fashion as inequalities 1–4, in the first step of the proof of this case. We now justify inequality 3. Note that $\omega' = \text{Run}(\omega_{i_j+2}, \mu'_\varepsilon, \chi)_{i_{j+1}-i_j-2}$ visits ℓ^* only upon entering $s_{i_{j+1}}$, hence $k = i_{j+1} - i_j - 2 \leq \min\{\text{Len}(\omega'), \text{Stop}_{\Gamma''}(\omega')\}$ — we can therefore apply Lemma 3.35. \square

Secondly we consider the case when $\ell \in \mathbf{L}^{\text{Min}}$.

Lemma 3.36. *Given a reachability-price game in which the control location with the smallest value of the price function is in \mathbf{L}^{Min} , if the third case of **SolveRP** procedure terminates, then it outputs Val_Γ , which is a cost function, locationwise.*

Proof. We will show that the single iteration of the third case of the algorithm correctly computes Val_Γ over the given interval. Without loss of generality, we assume that we are dealing with a single interval $I = [b, e]$ (in the third case of the algorithm it would have been $I_i = [b_i, e_i = r]$). Lemma 3.30 allows us to establish that results computed in subsequent operations are combined correctly.

Let ℓ^* , Γ' , Γ'' , and $x^* \in I$ be defined as in the third case of the algorithm. There are two cases; we will show that $\text{Val}_{\Gamma'}(s) = \text{Val}_{\Gamma_{[x^*, e]}}(s)$ for every state s , and that $\text{Val}_{\Gamma''}(s) = \text{Val}_{\Gamma_{[b, x^*]}}(s)$, for every state s . To achieve this goal, we will show that Val_Γ exists, i.e., that Γ is determined, and that that it is indeed equal to $\text{Val}_{\Gamma'}$ and $\text{Val}_{\Gamma''}$ over the designated sets of states. In particular, this implies that Val_Γ is a cost function, locationwise. In the proof, we will be relying on the inductive hypothesis that the games with one less non-urgent control location are determined; that the Val function for such games is a cost function, locationwise; and that the algorithm computes the game value correctly for such games. Note that Γ' and Γ'' have one less non-urgent control location than Γ .

The games Γ' and Γ'' can be simulated in Γ by choosing an appropriate strategy $\mu \in \Sigma^{\text{Min}}$. In the first case this strategy would spend all available time in ℓ^* , and in the second case this strategy would leave ℓ^* immediately. We use this observation to establish that, depending on the case:

$$\text{Val}_{\Gamma'}(s) \geq \text{Val}_\Gamma^*(s) \text{ or } \text{Val}_{\Gamma''}(s) \geq \text{Val}_\Gamma^*(s),$$

for every $s \in S$. Recall the definition of Val^* , which states that for every $\mu \in \Sigma^{\text{Min}}$, and every $s \in S$, we have that $\text{Val}^\mu(s) \geq \text{Val}_\Gamma^*(s)$.

To show determinacy of Γ , and to prove that Val_Γ is indeed computed correctly, it remains to show that, depending on the case, $\text{Val}_{*\Gamma}(s) \geq \text{Val}_{\Gamma'}(s)$ or $\text{Val}_{*\Gamma}(s) \geq \text{Val}_{\Gamma''}(s)$ for all states s . To achieve this, in the case of Γ' , we choose an $\varepsilon > 0$, and we fix a strategy $\chi_\varepsilon \in \Sigma^{\text{Max}}$, which has some desired properties, and show that for every $\mu \in \Sigma^{\text{Min}}$ and every state $s \in S$, we have that $\mathcal{P}_\Gamma(\text{Run}(s, \mu, \chi_\varepsilon)) \geq \text{Val}_{\Gamma'}(s) - \varepsilon$. This, due to the arbitrary choice of μ , establishes that $\text{Val}_{\Gamma_{\chi_\varepsilon}}(s) \geq \text{Val}_{\Gamma'}(s) - \varepsilon$, which in turn, by definition of Val_* , establishes that $\text{Val}_{*\Gamma}(s) \geq \text{Val}_{\Gamma'}(s)$, which yields the desired result. The case of Γ'' is handled analogously.

We will restrict the set of states to $(\{\ell^*\} \times I) \cap \mathbf{S}$, and apply Lemma 3.31 to establish that the result holds for all states.

There are two cases to consider. We start with the first case, i.e., we will show that $\text{Val}_{*\Gamma}(\ell^*)(x) \geq \text{Val}_{\Gamma'}(\ell^*)(x)$ for all $x \in [x^*, e]$.

Fix $\varepsilon > 0$ and $\chi_\varepsilon \in \Sigma^{\text{Max}}$, a ε -optimal strategy of player Max in the game Γ' (note that Γ and Γ' admit the same sets of strategies). For every strategy μ of player Min, and for every state $s \in (\{\ell^*\} \times [x^*, e]) \cap \mathbf{S}$, let ω denote the unique run in $\text{Run}(s, \mu, \chi_\varepsilon)$, and let's assume it visits ℓ^* exactly m times, after transitions i_1, \dots, i_m . We assume, without loss of generality, that after every such transition, a continuous one is taken. We then have:

$$\begin{aligned}
\mathcal{P}(\omega) &= \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_\Gamma(\text{Run}(\omega_{i_m+1}, \mu, \chi_\varepsilon)) \\
&\geq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_{\Gamma'}(\text{Run}(\omega_{i_m+2}, \mu, \chi_\varepsilon)) \\
&\geq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_m+2}) - \varepsilon \\
&\geq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_m+1}) - \varepsilon \\
&\geq \text{Price}(\omega_{i_1}) + (x_{i_m} + \lambda_{i_m+1} - x_{i_1}) \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}(s_{i_m+1}) - \varepsilon \\
&\geq \text{Price}(\omega_{i_1}) + (e - x_{i_1}) \cdot \pi(\ell^*) + \text{Val}_{\Gamma'}((\ell^*, e)) - \varepsilon \\
&= \text{Val}_{\Gamma'}(s) - \varepsilon,
\end{aligned}$$

where $s_i = (\ell_i, x_i)$. The first inequality holds because λ_{i_m+2} is a discrete transition, which does not incur a price, and because the suffix of ω , starting in s_{i_m+2} , does not visit ℓ^* . The second inequality holds because χ_ε is an ε -optimal strategy of player Max in Γ' . The third inequality holds because, by definition,

$$\text{Val}_{\Gamma'}(\ell^*)(x) \leq \min_{(\ell^*, \ell) \in \mathbf{A}'} \text{Val}_{\Gamma'}(\ell)(x)$$

over the interval $[x^*, e]$. The fourth inequality holds because ℓ^* minimises the value of the price function π . The fifth inequality holds because $s = s_{i_1}$, which implies

$\text{Len}(\omega) = 0$, and because, by definition, the slope of $\text{Val}_{\Gamma'}(\ell^*)$ is equal to $-\pi(\ell^*)$. The final equality follows from the definition of $\text{Val}_{\Gamma'}(\ell^*)$.

We now proceed to the second case, i.e., we will show that $\text{Val}_{*\Gamma}(\ell^*)(x) \geq \text{Val}_{\Gamma'}(\ell^*)(x)$ for $x \in [b, x^*]$.

Fix $\varepsilon > 0$ and $\chi_\varepsilon \in \Sigma^{\text{Max}}$, a ε -optimal strategy of player Max in the game Γ'' (note that Γ and Γ'' admit the same sets of strategies). For every strategy μ of player Min, and for every state $s \in (\{\ell^*\} \times [b, x^*]) \cap \mathbf{S}$, let ω denote the unique run in $\text{Run}(s, \mu, \chi_\varepsilon)$, and let's assume it visits ℓ^* exactly m times, after transitions i_1, \dots, i_m . We assume, without loss of generality, that after every such transition, a continuous one is taken. We then have:

$$\begin{aligned}
\mathcal{P}_\Gamma(\omega) &= \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_\Gamma(\text{Run}(\omega_{i_m+1}, \mu, \chi_{\varepsilon'})) \\
&\geq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \mathcal{P}_{\Gamma''}(\text{Run}(\omega_{i_m+2}, \mu, \chi_{\varepsilon'})) \\
&\geq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_m+2}, \mu, \chi_{\varepsilon'}) - \varepsilon \\
&\geq \text{Price}(\omega_{i_m}) + \lambda_{i_m+1} \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_m+1}, \mu, \chi_{\varepsilon'}) - \varepsilon \\
&\geq \text{Price}(\omega_{i_1}) + (x_{i_m} + \lambda_{i_m+1} - x_{i_1}) \cdot \pi(\ell^*) + \text{Val}_{\Gamma''}(s_{i_m+1}) - \varepsilon \\
&\geq \text{Price}(\omega_{i_1}) + \text{Val}_{\Gamma''}(s_{i_1}) - \varepsilon \\
&= \text{Val}_{\Gamma''}(s_{i_1}) - \varepsilon.
\end{aligned}$$

where $s_i = (\ell_i, x_i)$. The first inequality holds because λ_{i_m+2} is a discrete transition, which does not incur a price, and because the suffix of ω , starting in s_{i_m+2} , does not visit ℓ^* . The second inequality holds because χ_ε is an ε -optimal strategy of player Max in Γ'' . The third inequality holds and because

$$\text{Val}_{\Gamma''}(\ell^*)(x) = \min_{(\ell^*, \ell) \in \mathbf{A}''} \text{Val}_{\Gamma''}(\ell)(x)$$

over the interval $[x^*, e]$. The fourth inequality holds because ℓ^* minimises the value of the price function π . The fifth inequality holds because, by definition, the slope of

$\text{Val}_{\Gamma''}(\ell^*)$ is shallower than $-\pi(\ell^*)$. The final equality holds because $s = s_{i_1}$, which implies that $\text{Len}(\omega_{i_1}) = 0$. \square

We have proved that the algorithm is partially correct. It remains to prove its total correctness, i.e., that it terminates.

Theorem 3.37. *The SolveRP procedure terminates.*

Proof. To prove termination of the algorithm, we need to prove the termination of the iterative procedures from second and third cases of the algorithm.

Each of the two cases is different, however, they have one thing in common. In both cases, in each iteration an interval with slope shallower than $-\pi(\ell^*)$ is processed. Since ℓ^* minimises $\pi(\ell^*)$, by Proposition 3.18 and by Lemmas 3.17 and 3.26, this interval corresponds to a segment of a goal cost function. We argue that the number of iterations in each case is bounded by the number of all the possible intersections of the cost functions assigned to goal locations, which is finite.

More precisely, let Γ' , I , f and i be defined as in the second case of the algorithm. The slope of f_i over I_i is shallower than $-\pi(\ell^*)$, so by Lemmas 3.17 and 3.26, f_i coincides with some cost function over I_i — denoted by g . If $i > 1$, then there are two possibilities, either b_i coincides with an intersection of cost functions from two different goal locations, or otherwise f_{i-1} has the slope equal to $-\pi(\ell)$ for some non-goal control location ℓ .

In the first case, the iteration processed one of the finitely many intersection points. In the second case, we need to argue that if the procedure processes the same affine piece of g more than once (but over a different interval), then it must have also processed at least one of the finitely many intersection points. Let $I' = [b', e'] \subseteq [b, b_i)$ denote the interval over which the procedure encounters the same affine piece of g again. Assume that the procedure did not process any intersection points before I' . This implies that $\text{Val}_{\Gamma'}(\ell^*) \geq g$ over $[e', b_i]$, and hence $\text{Val}_{\Gamma'}(\ell^*)$ must contain an affine piece that has a slope shallower than g over $[e', z]$, for some

$z \in (e', b_i]$. However, such a piece coincides with a piece of a goal cost function, so an intersection point must have been processed.

So far we have shown termination of the iterative procedure in the second case of the algorithm. It remains to show the same for the third case of the algorithm. Let Γ' , f , i , and x^* be defined as in the third case of the algorithm. We argue that each affine segment of a goal cost function is processed only once. If $i > 1$ then f_{i-1} either coincides with a different piece of a goal cost function, which means we have processed one of the finitely many intersection points and the f_i segment has been processed, or its slope is equal to $-\pi(\ell)$ for some non-goal control location ℓ . In the latter case we have that $\text{Val}_{\Gamma'}(\ell^*)$ has a slope steeper than f_i , and hence, it is strictly greater than f_i over $[b, b_i)$. This means that in the subsequent iterations, if f_i is to be processed once again (but over a different interval), a piece with a slope smaller than that of f_i needs to occur. However, such a piece would coincide with a different segment of an outside cost function, which means that an intersection point would be processed prior to processing f_i once again.

We have shown that in each step of the algorithm the iterative procedure of the second and third case of the algorithm terminates, and hence, the algorithm terminates. \square

We have proved that the **SolveRP** procedure is correct. The question that remains is its complexity. We have the following result.

Theorem 3.38. *The **SolveRP** procedure terminates in exponential time.*

Proof. Given an automaton \mathcal{A} , let n denote the number of non-urgent control locations and p the total number of pieces in CF^F . The complexity of computing the solution, using **SolveRP** procedure, depends on the number of affine pieces that constitute Val_{Γ} . Let $N(n, p)$ denote the upper bound on the number of pieces in Val_{Γ} . We now construct a recursion to characterise $N(n, p)$.

If $n = 0$ then $N(n, p) \leq 2p$ (recall Fact 3.15. If $n > 0$ then both cases take p

(as argued in the proof of Theorem 3.37) iterations, and each requires solving two games whose Val functions comprise of at most $N(n-1, p+n)$ and $N(n-2, p+n-1)$ affine segments, respectively. We can assume that $p > n$ is the case of real interest, so we have $N(n, p) \leq 2pN(n-1, 2p)$. It can be easily verified that:

$$N(n, p) \leq 2^{\frac{(n+1)(n+2)}{2}} p^{n+1}.$$

One can easily check that this is the case for $n = 0$. Assume that it holds for some $n \geq 0$, we have:

$$\begin{aligned} N(n+1, p) &\leq 2p \cdot N(n, 2p) \\ &\leq 2p \cdot 2^{\frac{(n+1)(n+2)}{2}} \cdot (2p)^{n+1} \\ &\leq 2^{\frac{(n+1)(n+2)}{2} + n+2} p^{n+2} \\ &\leq 2^{\frac{(n+2)(n+3)}{2}} p^{n+2}. \end{aligned}$$

This establishes that the number of affine pieces that constitutes the Val_Γ function is at most exponential in the size of the game Γ , which in turn establishes that the procedure SolveRP terminates in exponential time. \square

Chapter 4

Games on price-reward graphs

Price-reward game graphs generalise finite doubly-weighted game graphs, as presented in Section 2.2.2. A finite doubly-weighted game graph is a directed graph in which the set of states is partitioned between the two players, Min and Max. The two weights assigned to edges are the price and the reward. The extension of doubly-weighted game graphs presented in this chapter is obtained by adding complexity to the pricing (rewarding) mechanism.

We obtain a price-reward game graph by adding a dynamic weighing mechanism — the prices (rewards) are no longer predetermined. Each edge is assigned two sets of inputs, one for each of the two players. Every time an edge is traversed, players choose their inputs to determine the edge's price and reward. Additionally, price-reward game graphs, as presented in this chapter, may be infinite.

As models, price-reward game graphs allow for modelling systems that feature transitions whose price and reward is dependent on the behaviour of the controller and the environment. Notice that this behaviour affects only the quantitative aspect of the transition, not its initial or final state.

This chapter is organised as follows: first we introduce price-reward game graphs and price-reward games; then we proceed to define average-price-per-reward games on these graphs, followed by an optimality equation characterisation of game

values; we conclude the chapter with a proof that average-price-per-reward games, on finite price-reward game graphs, are determined.

This determinacy result is the main result of this chapter, and is obtained through a “strategy improvement” like argument. Later, in Chapter 5, we will show that we can reduce average-price-per-reward games on hybrid systems with strong resets to average-price-per-reward games on finite price-per-reward games.

4.1 Price-reward game graphs

We start this section by defining price-reward game graphs, and then we proceed to introduce the standard definitions of runs, strategies, payoffs, and game values which are needed to formally define price-reward games on price-reward game graphs. The definitions presented here are similar in spirit to those found Section 2.2.2, but account for the more complex player interaction. We present them in full for the sake of clarity.

Definition 4.1 (Price-reward game graph). *A price-reward game graph Γ is given by*

$$\Gamma = \langle S^{\text{Min}}, S^{\text{Max}}, E, \mathcal{I}, \Theta^{\text{Min}}, \Theta^{\text{Max}}, \pi, \rho \rangle,$$

where:

- $S^{\text{Min}} \cap S^{\text{Max}} = \emptyset$ are the (possibly infinite) sets of states for players Min and Max respectively,
- $\langle S^{\text{Min}} \cup S^{\text{Max}}, E \rangle$ is a (possibly infinite) directed graph,
- \mathcal{I} is the set of inputs,
- $\Theta^{\text{Min}} : E \rightarrow \mathcal{P}(\mathcal{I})$ is player Min’s input function,
- $\Theta^{\text{Max}} : E \rightarrow \mathcal{P}(\mathcal{I})$ is player Max’s input function,
- $\pi : E \times \mathcal{I}^2 \rightarrow \mathbb{R}$ is the price function,

- $\rho : E \times \mathcal{I}^2 \rightarrow \mathbb{R}$ is the reward function.

Remark 4.2. *The definition of a price-reward game graph does not place any restrictions on the price and reward functions. However, as it will be explained in Section 4.1.2, some global properties, such as divergence and boundedness, will be required. They are omitted from the definition, as they are semantic rather than syntactic.* \square

A price-reward game graph Γ is said to be *definable* if all its components are definable. When the payoff functions are given, we will refer to Γ as a *price-reward game*.

Similarly to the finite game graphs introduced in Section 2.2.2 the game is played by moving a token from one state to another, with the state owner deciding which edge to take. In this case, however, once an edge is chosen, both players choose inputs, which determine the price and reward incurred.

Remark 4.3. *If, for every edge, the price and reward are independent of the inputs chosen, the inputs can be omitted from the specification. In such a case, we obtain the classical definition of a doubly-weighted game graph.* \square

The following example shows, how the notion of inputs can be used in modelling.

Example 4.4. *Arbitrage is a practise of gaining profit by exploiting the difference of asset valuation between one or more markets. Figure 4.1 models a very simple arbitrage process. There are two markets: $M1$, and $M2$, and each of them can be in one of the two states of the conjunctural cycle: High (H) and Low (L).*

Player Min models a trader, who continuously buys and sells an asset. Buying happens when she chooses to proceed to state “Bought”. Upon choosing the relevant edge, both players choose the inputs, which determine the price and reward of the edge. Player Min chooses the market in which to buy the asset, and player Max,

who models the environment, chooses the state of the conjunctural cycle. Similar interaction occurs when player Min decides to sell, by proceeding to the state “Sold”.

Consider the situation where Min does not hold any assets. She proceeds to buy, and chooses market M1; player Max decides that the markets are currently at the high state of the conjunctural state. The price and reward of this edge are thus 5 and 1 respectively.

After a while, player Min decides to sell the assets. She chooses to sell at market M2, but unfortunately there is a downward swing in the market conjuncture, which is reflected by Max’s choice of Low conjunctural state. The price and reward of the relevant edge are thus 1 and 2.

In this particular scenario, player Min lost on her arbitrage attempt; she paid 6 and her reward was only 3. \square

We now introduce some basic definitions for price-reward game graphs. We start with moves and runs, and then proceed to strategies. The concepts are introduced in the usual way (see Section 2.2.2), with the only difference that we account for inputs. Moves are labelled with chosen inputs¹, and strategies have two components: for “choosing edges” and for “choosing edge inputs”.

We write $s \xrightarrow{\theta} s'$ to denote a move, where $e = (s, s') \in E$ and $\theta \in \Theta^{\text{Min}}(e) \times \Theta^{\text{Max}}(e)$. The price of the move is $\pi(e, \theta)$ and the reward is $\rho(e, \theta)$. A run is a (possibly infinite) sequence of moves $\omega = s_0 \xrightarrow{\theta_1} s_1 \xrightarrow{\theta_2} s_2 \dots$. The set of all runs of Γ is denoted by Runs , and its subset of all *finite runs* by Runs_{fin} .

A *state strategy* of player Min is a partial function $\mu^S : \text{Runs}_{\text{fin}} \rightarrow E$ which is defined on all runs ending with some $s \in S^{\text{Min}}$. A strategy is called *positional* if for every two runs $\omega, \omega' \in \text{Runs}_{\text{fin}}$ we have, if ω and ω' end in the same state then $\mu(\omega) = \mu(\omega')$, i.e., it can be viewed as a function $\mu^S : S^{\text{Min}} \rightarrow E$. Given an edge e , an *e-strategy* of player Min is an element $x \in \Theta^{\text{Min}}(e)$. An *edge strategy* $\mu^E :$

¹Note that in this context the labels assigned to moves differ from the labels in a labelled transition relation. There the label is a part of a transition, whereas here the label is dynamic and depending on the players choices, the same transition may bear different labels.

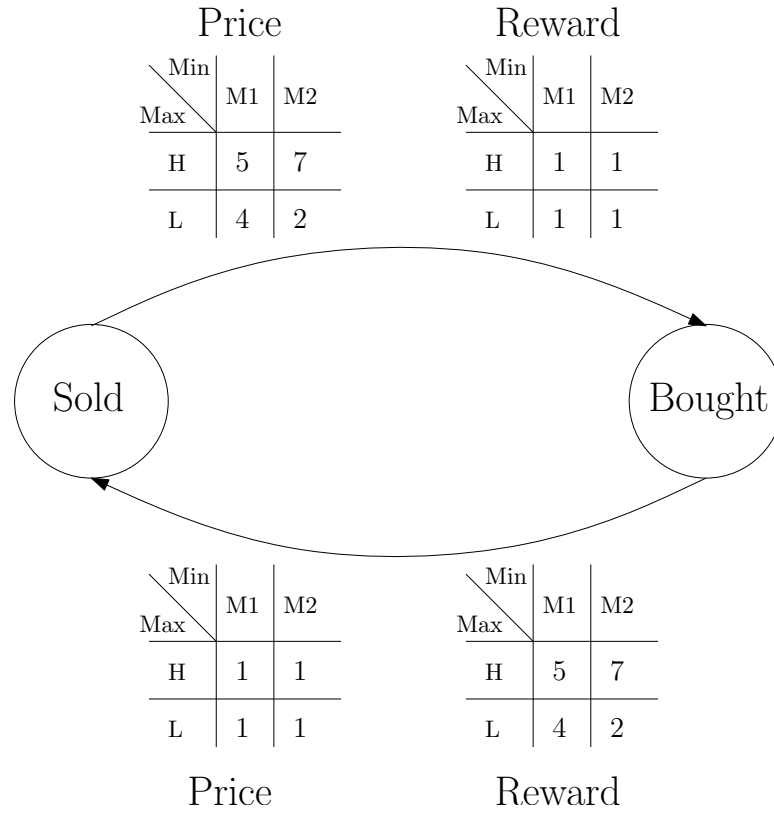


Figure 4.1: A simple price-reward game graph modelling arbitrage between two markets. All states belong to player Min. The matrices annotating the edges show how the price (reward) of an edge changes depending on players' inputs. Player Min can choose between M1 and M2, whereas player Max can choose between L and H.

$\text{Runs}_{\text{fin}} \times \mathbf{E} \rightarrow \mathcal{I}$ of player Min is a partial function, defined on all pairs of finite runs ω and edges e such that e originates from the ending state of ω , that assigns an e -strategy. Similarly to state strategies, an edge strategy is called positional if for every two runs $\omega, \omega' \in \text{Runs}_{\text{fin}}$, if ω and ω' end in the same state then for every edge e originating in that state we have $\mu(\omega, e) = \mu(\omega', e)$, i.e., $\mu^{\mathbf{E}}$ can be viewed as a function $\mu^{\mathbf{E}} : \mathbf{E} \rightarrow \mathcal{I}$.

A strategy μ of player Min is a pair $(\mu^{\mathbf{S}}, \mu^{\mathbf{E}})$ of state and edge strategies. We denote the set of all such strategies by Σ^{Min} . We say that μ is positional if both $\mu^{\mathbf{S}}$ and $\mu^{\mathbf{E}}$ are positional. We denote the set of all such positional strategies by Π^{Min} . The definitions for player Max are analogous.

Given strategies μ and χ of players Min and Max, and a state s , we write $\text{Run}(s, \chi, \mu)$ to denote the run starting in s that results from the players playing according to μ and χ .

Price-reward games. The definition of a game on a price-reward game graph is complete once we have supplied a payoff function. In this context, the payoff function maps a run of the game to a real value.

Definition 4.5 (Price-reward game). *A price-reward game is a pair $\langle \Gamma, \mathbf{P} \rangle$ where:*

- Γ is a price-reward game graph, and
- $\mathbf{P} : \text{Runs} \rightarrow \mathbb{R}$ is a payoff function.

When the payoff function is clear from the context, we will omit it and refer to Γ as the price-reward game.

Note that we do not mention definability of \mathbf{P} . This is because, in most cases, it will not be definable. However, that will not be a problem, as we will not need to approximate or decide the value of this function.

We can now introduce the concept of the value of a price-reward game from a state. We do this by defining a zero-sum game in strategic form (recall Section 2.2.1),

which is specific to the given state. Note that game value is in fact a partial function from states to real values.

Given a state s , let $\mathcal{D}_s = \langle \Sigma^{\text{Min}}, \Sigma^{\text{Max}}, \mathbf{P}(\text{Run}(s, \cdot, \cdot)) \rangle$ be a zero-sum game in strategic form. We say that the game Γ is *determined* from s if \mathcal{D}_s is determined, i.e., $\text{Val}_*(\mathcal{D}_s) = \text{Val}^*(\mathcal{D}_s)$, and *positionally determined* if

$$\text{Val}(\mathcal{D}_s) = \inf_{\mu \in \Pi^{\text{Min}}} \text{Val}^\mu(\mathcal{D}_s) = \sup_{\chi \in \Pi^{\text{Max}}} \text{Val}_\chi(\mathcal{D}_s).$$

We say that the price-reward game Γ is determined if it is determined from every state.

For simplicity we will write $\text{Val}(s)$ rather than $\text{Val}(\mathcal{D}_s)$, in the context of price-reward games, so Val can be viewed as a partial function $\mathbf{S} \rightarrow \mathbb{R}$.

We will say that a price-reward game Γ is *decidable* if the partial function $\text{Val} : \mathbf{S} \rightarrow \mathbb{R}$ is decidable (recall Definition 2.17). We emphasise that Val is a partial function because Γ does not have to be determined from every state.

4.1.1 Average-price-per-reward games

In this section, we introduce average-price-per-reward games, and provide a characterisation of game values using a set of equations, referred to as average-price-per-reward optimality equations. The key result is Theorem 4.10, which states that solutions to average-price-per-reward optimality equations coincide with game values.

The results presented here are general, and will be applied to finite average-price-per-reward games (Section 4.2) as well as to their hybrid counterparts (Section 5.1.1). The fact that, in both cases, the game values are characterised using average-price-per-reward optimality equations will be used in the proof of the reduction from hybrid games to finite games (Section 5.2.2).

An *average-price-per-reward game* is played on a price-reward graph Γ . This

game will be the focus of the rest of the chapter. We will abuse the notation, as explained earlier, and simply write Γ , without specifying the payoff functions, to denote the average-price-per-reward game.

The goal of player Min in an average-price-per-reward game Γ is to minimise the average price-over-reward ratio in a run, and the goal of player Max is to maximise it.

As explained in Section 2.2.2 we need to resort to lower and upper payoff functions to properly define an average-price-per-reward game. This is not a problem, as strategy and game values for a state of Γ are defined using the framework of zero-sum games in strategic form.

We define the upper and lower payoff functions in the following way:

$$\mathcal{A}^*(\omega) = \limsup_{n \rightarrow \infty} \frac{\sum_{i=0}^n \pi(e_{i+1}, \theta_{i+1})}{\sum_{i=0}^n \rho(e_{i+1}, \theta_{i+1})}$$

and

$$\mathcal{A}_*(\omega) = \liminf_{n \rightarrow \infty} \frac{\sum_{i=0}^n \pi(e_{i+1}, \theta_{i+1})}{\sum_{i=0}^n \rho(e_{i+1}, \theta_{i+1})},$$

where ω is an infinite run, $s_i \xrightarrow{\theta_{i+1}} s_{i+1}$ and $e_{i+1} = (s_i, s_{i+1})$ for all $i \geq 0$. Note that $\mathcal{A}_* \leq \mathcal{A}^*$, as required in Remark 2.11.

To guarantee that the payoffs, as introduced above, are always well-defined we introduce the notions of *reward divergence*, and *price* and *reward boundedness*.

Definition 4.6 (Reward divergence). *We say that Γ is $f(n)$ -reward divergent with a constant $c > 0$ if, for every run ω , there exists $N \in \mathbb{N}$ such that*

$$\sum_{i=0}^n \rho(e_{i+1}, \theta_{i+1}) \geq c \cdot f(n),$$

for all $n \geq N$.

We assume that Γ is n -reward divergent. Linear (i.e., n) reward divergence

is required in the proof of Theorem 4.10. Notice that in this definition c is global whereas N is local, i.e., depends on a particular run. In the proofs of Lemmas 4.12–4.13, existence of global c enables us to define ε -positional strategies, however, to prove that this definition is correct, existence of local N suffices.

Additionally, we require that Γ is both *price* and *reward bounded*, i.e., there exists $M \in \mathbb{R}_{>0}$ such that $|\pi(e, \theta)| < M$ and $|\rho(e, \theta)| < M$ for all $e \in E$, and all $\theta \in \mathcal{I}^2$. This is necessary to ensure that edge games, as introduced below, are determined. Moreover, without loss of generality, we assume that the games are non-blocking, i.e., that for every state s there exists a state s' such that $(s, s') \in E$.

The divergence requirement can be seen as a generalisation of the non-Zenoness requirement to rewards (as in the work of Bouyer et al. [BBL08]); we want to prevent runs that admit finite rewards. We need at least linear divergence in the proof of Theorem 4.10. Note that if the reward is simply time, then it is equivalent to the non-Zenoness condition, i.e., that time diverges. Also note that one can guarantee n -reward divergence by choosing ρ with a strictly positive lower bound, i.e., such that there exists $c > 0$ and $\rho(e, \theta) > c$, for all $e \in E$ and all $\theta \in \mathcal{I}^2$.

4.1.2 Optimality equations

Proving that game values exist and computing them using their definitions directly is difficult. Moreover, as mentioned earlier, the payoff function is not definable, which renders our computational framework useless. In this part of the chapter, we will introduce a way to alleviate both of those problems.

Similarly to Section 2.2.2, we consider two functions $G, B : S \rightarrow \mathbb{R}$. We will refer to these functions as *gain* and *bias* respectively, and formulate a set of equations, called average-price-per-reward optimality equations. Those equations have the property that if gain and bias functions satisfy those equations, then $G(s) = \text{Val}(s)$ for all $s \in S$.

Optimality equations can be seen as a characterisation of game values, and

their solutions, if they exist, as witnesses to existence of positional ε -optimal strategies for both players. In the book by Puterman [Put94], a variant is used in the setting of discounted Markov decision problems. On the other hand, one can also interpret optimality equations as a medium for reducing an infinite duration game to a one step game. The latter interpretation comes from the fact that optimality equations characterise globally optimal choices using local conditions.

To introduce average-price-per-reward optimality equations, we need to define a special *edge game*. This new game will be using the edge input sets as strategy sets, and the payoff function will be given by an algebraic expression that involves the price and reward functions, as well as a special parameter g . The notion of an edge game is used for technical reasons, however, it also provides insight into the interpretation of average-price-per-reward optimality equations, especially in the case of hybrid games, introduced in Chapter 5.

When optimising a move in an average-price-per-reward game, one cannot decide the best choice of inputs locally, i.e., solely on the values of price and reward. The special parameter g can therefore be seen as the carrier of information about the global effect of the move. We will see that, if for every edge players play “optimally” in this newly introduced edge game, then their play is globally optimal, for the given choices of edges.

Let Γ be an average-price-per-reward game. For every edge e , we introduce a game

$$\mathfrak{D}_e(g) = \langle \Theta^{\text{Min}}(e), \Theta^{\text{Max}}(e), P_e(g) \rangle,$$

where g is a real-valued parameter, and $P_e(g) : \mathcal{I}^2 \rightarrow \mathbb{R}$ is defined as $\pi(e, \cdot, \cdot) - \rho(e, \cdot, \cdot) \cdot g$. We will refer to it as an *edge game*. In the formulation of average-price-per-reward optimality equations we will be referring to the values of the edge game, so it is necessary to assume that these games are determined, regardless of the value of g . Note that, for every $e \in E$ and $g \in \mathbb{R}$, we have that $\mathfrak{D}_e(g)$ is definable if Γ is

definable.

Assumption 4.7 (Edge game determinancy). *For the reminder of this section, we assume that the price-per-reward game Γ is such that, for every edge e , and every real value g the game $\mathcal{D}_e(g)$ is determined in pure strategies.* \square

Let $G, B : \mathcal{S} \rightarrow \mathbb{R}$ such that the range of G is finite, and B is bounded. We say that a pair of functions (G, B) is a solution of *average-price-per-reward optimality equations* for Γ , denoted by $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$, if the following conditions hold for all states $s \in \mathcal{S}^{\text{Min}}$:

$$G(s) = \min\{G(s') : (s, s') \in \mathcal{E}\}, \quad (4.1)$$

$$B(s) = \inf\{\text{Val}(\mathcal{D}_{(s,s')}(G(s'))) + B(s') : (s, s') \in \mathcal{E} \text{ and } G(s) = G(s')\}; \quad (4.2)$$

and if the following two equations hold for all states in \mathcal{S}^{Max} :

$$G(s) = \max\{G(s') : (s, s') \in \mathcal{E}\}, \quad (4.3)$$

$$B(s) = \sup\{\text{Val}(\mathcal{D}_{(s,s')}(G(s'))) + B(s') : (s, s') \in \mathcal{E} \text{ and } G(s) = G(s')\}. \quad (4.4)$$

Note that we write min and max, instead of inf and sup, in the equations 4.1 and 4.3, respectively. We can do that because we assumed that the range of the gain function is finite. In finite price-reward game graphs for instance, due to the finite number of edges, the gain function has to have a finite range.

Remark 4.8. *If Γ is definable then $\text{Opt}_{\text{AvgPR}}(\Gamma)$ is first-order expressible in \mathcal{M} .* \square

In the following example, we show why the edge games, as used in the formulation of the average-price-per-reward optimality equations, have to be parametrised.

Example 4.9. *Figure 4.2 depicts a simple, one player average-price-per-reward game. Note that, depending on the state strategy chosen, a different edge strategy is beneficial for player Min.*

Consider the edge game $\mathfrak{D}(g)$ corresponding to the horizontal edge. Player Min has two inputs to choose from, for simplicity we refer to them as “left” and “right”. Given a parameter g , the payoff function is defined as follows:

$$P(g)(i) = \begin{cases} 2 - 0 \cdot g & \text{if } i = \text{left} \\ 0 - 2 \cdot g & \text{if } i = \text{right} \end{cases}$$

Depending on the g chosen, a different input is optimal. For instance, as Figure 4.2 shows, we have:

$$P\left(-\frac{1}{3}\right)(\text{left}) > P\left(-\frac{1}{3}\right)(\text{right}),$$

and

$$P(-2)(\text{left}) < P(-2)(\text{right}).$$

Consider the average-price-per-reward optimality equations for the game in Figure 4.2. In every solution, to the optimality equations, the value of gain, for every state, is equal to -2 . Notice that, the optimal input in the edge game $\mathfrak{D}(-2)$, “left”, is in fact optimal in the whole game, when Min plays optimally in the grey state, i.e., by playing “up”. \square

We now present the main result of this section, i.e., that equations 4.1–4.4 indeed characterise the game values of a average-price-per-reward games on price-reward game graphs.

Theorem 4.10. *If $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$ then for every state s , the average-price-per-reward game Γ from s is determined and we have $\text{Val}(s) = G(s)$. Moreover, for every $\varepsilon > 0$, positional ε -optimal strategies exist for both players.*

Proof. The theorem follows from Lemmas 4.12 and 4.13, and their proofs. The lemmas imply that for all states s , we have $\text{Val}^*(s) \leq G(s)$ and $\text{Val}_*(s) \geq G(s)$. This in turn implies that $\text{Val}^*(s) = \text{Val}_*(s) = G(s)$. Moreover, the proofs of the lemmas are in fact constructive proofs of the existence of ε -optimal strategies. \square

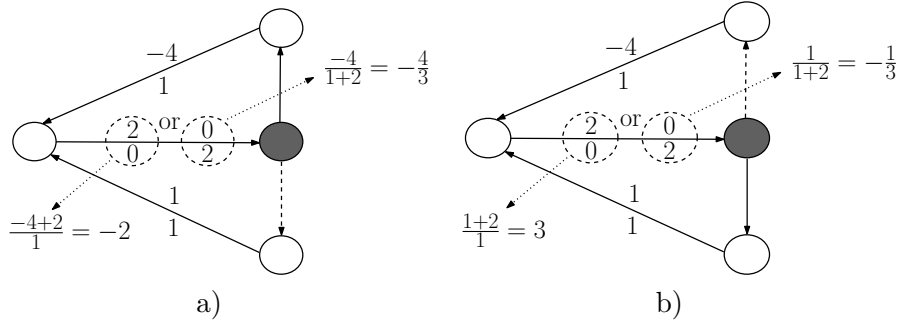


Figure 4.2: A simple price-reward graph; all edges, except the horizontal one, have fixed prices and rewards. The price of an edge is written above it, whereas the reward is written below. The horizontal edge admits two inputs from player Min, and the set of inputs for player Max is empty. All states belong to player Min. Figure a) shows the values of the price-per-reward game from every state, depending on the chosen inputs, when player Min chooses to play “up” from the grey state. Likewise, Figure b) shows the respective values when player Min chooses to play “down” from the grey state.

Corollary 4.11. *If there exists definable (G, B) such that $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$ and Γ is definable, then positional ε -optimal strategies are definable.*

Proof. Positional strategies, as constructed in the proof of Lemmas 4.12–4.13, are definable if (G, B) , the solution to $\text{Opt}_{\text{AvgPR}}(\Gamma)$, is definable. \square

Lemma 4.12. *Let $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$. Then for all $\varepsilon > 0$, there is $\mu_\varepsilon \in \Pi_{\text{Min}}$ such that for all $\chi \in \Sigma^{\text{Max}}$ and for all $s \in S$, we have $\mathcal{A}^*(\text{Run}(s, \mu_\varepsilon, \chi)) \leq G(s) + \varepsilon$.*

Lemma 4.13. *Let $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$. Then for all $\varepsilon > 0$, there is $\chi_\varepsilon \in \Pi_{\text{Max}}$ such that for all $\mu \in \Sigma^{\text{Min}}$ and for all $s \in S$, we have $\mathcal{A}_*(\text{Run}(s, \mu, \chi_\varepsilon)) \geq G(s) - \varepsilon$.*

We omit the proof of Lemma 4.13 as it is similar to the proof of Lemma 4.12.

Proof of Lemma 4.12. As argued earlier, a solution to average-price-per-reward optimality equations can be seen as a witness for existence of positional ε -optimal strategies. We will now show how to use this solution, if it exists, to construct an ε -optimal strategy for player Min. The construction is fairly technical and can be seen to be focused on choosing locally optimal moves.

Recall that Γ is n -divergent with constant c . Let us fix $\varepsilon > 0$ for the rest of the proof.

Given an edge e , $\varepsilon' > 0$ and $g \in \mathbb{R}$ let $x(e, \varepsilon', g) \in \Theta^{\text{Min}}(e)$ denote the ε' -optimal strategy of player Min in the game $\mathcal{D}_e(g)$, i.e.,

$$\text{Val}^{x(e, \varepsilon', g)}(\mathcal{D}_e(g)) \leq \text{Val}(\mathcal{D}_e(g)) + \varepsilon'.$$

We will now construct a positional strategy $\mu_\varepsilon = (\mu_\varepsilon^S, \mu_\varepsilon^E)$ for player Min and then show that it is ε -optimal. We begin by defining an edge strategy μ_ε^E . For every edge $e = (s, s')$ we require that:

$$\mu_\varepsilon^E(e) = x\left(e, \frac{c \cdot \varepsilon}{2}, G(s)\right)$$

If the solutions to the average-price-per-reward optimality equations exist and edge games are determined, this requirement can be satisfied. We now proceed to defining a state strategy μ_ε^S . For every state s , and $\mu_\varepsilon^S(s) = s'$ we require that:

$$\begin{aligned} G(s) &= G(s') \\ B(s) &\geq \text{Val}(\mathcal{D}_e(G(s)) + B(s') - \frac{c \cdot \varepsilon}{2} \end{aligned}$$

where $e = (s, s')$. Again, if average-price-per-reward optimality equations solutions exist and edge games are determined, this requirement can be satisfied. Given the earlier definition of μ_ε^E , for every $s \in S^{\text{Min}}$ the following holds:

$$\begin{aligned} G(s) &= G(s') \\ B(s) &\geq P_e(G(s))(\mu_\varepsilon^E(e), y) + B(s') - c \cdot \varepsilon, \end{aligned}$$

for all $y \in \Theta^{\text{Max}}(e)$.

We now proceed to prove, that μ_ε defined in this way is indeed ε -optimal

for player Min. For that purpose, let us fix a state s and some arbitrarily chosen strategy of player Max, $\chi \in \Sigma^{\text{Max}}$. Let $s_i \xrightarrow{\theta_{i+1}} s_{i+1}$ be the $(i+1)$ -th move of $\text{Run}(s, \mu_\varepsilon, \chi)$, and let $e_{i+1} = (s_i, s_{i+1})$.

From the definition of μ_ε , regardless of χ , we have:

$$G(s_i) \geq G(s_{i+1})$$

Therefore, since the range of G is finite, there exists a $K \in \mathbb{N}$ such that for all $i \geq K$, we have $G(s_i) = G(s_K)$. We will use $g = G(s_K)$ in the rest of the proof.

Let $L > K$. Our definition of μ_ε assures, that regardless of χ , for all $i = K, \dots, L-1$, the following holds:

$$B(s_i) \geq P_{e_{i+1}}(g)(\theta_{i+1}) + B(s_{i+1}) - c \cdot \varepsilon.$$

If we sum up the $L - K$ inequalities (recall that $P_e(g)(\theta) = \pi(e, \theta) - \rho(e, \theta) \cdot g$), we get:

$$\sum_{i=K}^{L-1} B(s_i) \geq \sum_{i=K+1}^L \pi(e_i, \theta_i) - g \cdot \sum_{i=K+1}^L \rho(e_i, \theta_i) + \sum_{i=K+1}^L B(s_i) - (L - K) \cdot c \cdot \varepsilon$$

That simplifies to:

$$\frac{B(s_K) - B(s_L)}{\sum_{i=K+1}^L \rho(e_i, \theta_i)} + g \geq \frac{\sum_{i=K+1}^L \pi(e_i, \theta_i) - (L - K) \cdot c \cdot \varepsilon}{\sum_{i=K+1}^L \rho(e_i, \theta_i)}$$

Recall that B is bounded, which implies that the left hand side converges to g as $L \rightarrow \infty$. Due to n -reward divergence of Γ (with a constant c), there exists an N such that for all $L > N$ we have:

$$\frac{(L - K) \cdot c \cdot \varepsilon}{\sum_{i=K+1}^L \rho(e_i, \theta_i)} \leq \varepsilon.$$

If we take L to the limit we obtain the following:

$$g \geq \mathcal{A}^*(\text{Run}(s, \mu_\varepsilon, \chi)) - \varepsilon$$

This, due to the arbitrary choice of s and χ , yields the desired result. \square

4.2 Finite average-price-per-reward games

In this section we state and prove the main results of this chapter, i.e., that finite average-price-per-reward games are determined (Theorem 4.14) and decidable². We start by showing determinacy of average-price-per-rewards on finite price-reward game graphs of out-degree one, i.e., game graphs in which the players only decide on their edge strategies. To establish this result we use the fact that game values are characterised using average-price-per-reward optimality equations. We conclude this section by using strategy improvement to establish determinacy in the general case.

The rest of this section is committed to the proof of Theorem 4.14. Its proof follows from Corollary 4.20 and Theorem 4.10, and Theorem 4.22.

Theorem 4.14. *Average-price-per-reward games on finite price-reward graphs are positionally determined and decidable.*

To guarantee uniqueness of the constructions, and for technical convenience, we fix a linear order on the states of the game graph. Given a subgraph $S \subseteq \Gamma$, $\min(S)$ denotes the smallest state in S .

4.2.1 Finite game graphs of out-degree one

We prove the following: average-price-per-reward games on finite price-per-reward game graphs of out degree one are determined. We start by introducing some

²By finite, we mean that the directed graph $\langle S, E \rangle$ is finite.

auxiliary notation regarding strategy subgraphs, and then proceed to establish the aforementioned result.

Strategy subgraphs. Let Γ be a finite price-reward game graph. Let μ^S be a positional state strategy for player Min. Such a strategy induces a subgraph of Γ , where the E relation is substituted by E_μ defined as

$$E_\mu = \{(s, s') : s \in S^{\text{Min}} \text{ and } \mu^E(s) = s', \text{ or } s \in S^{\text{Max}}\}.$$

We denote this game graph by Γ_{μ^S} . For simplicity, we will write $\Gamma_{\mu^S \chi^S}$ instead of $(\Gamma_{\mu^S})_{\chi^S}$.

A finite connected price-reward game graph of out-degree one is called a *sun*. Such a graph contains a unique cycle, referred to as the *rim*. States which are on the rim are called *rim states* and the remaining ones are called *ray states*.

Remark 4.15. If $\mu \in \Pi^{\text{Min}}$, $\chi \in \Pi^{\text{Max}}$, and Γ is a price-reward game graph, then $\Gamma_{\mu^S \chi^S}$ is a union of suns with disjoint rims. \square

Game graphs of out-degree one. In price-reward game graphs of out-degree one, strategies of both players are reduced to edge-strategies only. Without loss of generality, we can assume that the finite price-reward game Γ is defined on a single sun. We now provide a characterisation of upper and lower game values using the values of the rim edge games.

Lemma 4.16. Let Γ be finite average-price-per-reward game defined on a sun, and let e_1, \dots, e_k denote the edges that form the rim of that sun. Given a parameter $p \in \mathbb{R}$, the following is true for every state s :

- If $\sum_{i=1}^k \text{Val}(\partial_{e_i}(p)) \geq 0$, then $p \leq \text{Val}_*(s)$,
- If $\sum_{i=1}^k \text{Val}(\partial_{e_i}(p)) \leq 0$, then $p \geq \text{Val}^*(s)$.

Strict inequalities on the left hand side imply strict inequalities on the right hand side.

Proof. The proof is similar to that of Lemmas 4.12 and 4.13. We only sketch the proof of the first statement, as the other is symmetric.

Let χ be a strategy of player Max such that it is $c \cdot \varepsilon$ -optimal for every edge game $\mathcal{D}_{e_i}(p)$, for some $\varepsilon > 0$ and $i = 1, \dots, k$. If μ is a strategy of player Min, then for every edge e_i :

$$\pi(e_i, \chi(e_i), \mu(e_i)) - \rho(e_i, \chi(e_i), \mu(e_i)) \cdot p + c \cdot \varepsilon \geq \text{Val}(\mathcal{D}_{e_i}(p))$$

If we add up the k inequalities, we get:

$$\sum_{i=1}^k \pi(e_i, \chi(e_i), \mu(e_i)) - \sum_{i=1}^k \rho(e_i, \chi(e_i), \mu(e_i)) \cdot p + k \cdot c \cdot \varepsilon \geq 0$$

which gives:

$$\frac{\sum_{i=1}^k \pi(e_i, \chi(e_i), \mu(e_i))}{\sum_{i=1}^k \rho(e_i, \chi(e_i), \mu(e_i))} + \varepsilon \geq p$$

This, due to the arbitrary choice of ε and μ , finishes the proof. \square

Theorem 4.17. *On finite graphs of out-degree one, solutions to average-price-per-reward optimality equations exist.*

Proof. Let Γ be a finite average-price-per-reward game on a graph of out-degree one, and let S be one of the suns. For every state, both the upper and lower values are finite (recall that Γ is price-reward bounded and linearly reward divergent). Using binary search, together with Lemma 4.16, it follows that they are indeed equal.

Let g be the value of the game on sun S . We set the gain of all states to g , and the bias of $\min(S)$ to zero. The bias of the remaining states is set to the weight of the shortest path to $\min(S)$, assuming $\text{Val}(\mathcal{D}_e(g))$ to be the weight on the edge e . Gain and bias defined in this way satisfy average-price-per-reward optimality

equations. □

4.2.2 The general case

We have proved that games on graphs of out-degree one are determined. We will now use this result to prove determinacy in the general case.

When both players fix their positional state strategies, the game becomes a game on a graph of out degree one. We already know that we can define the solutions of average-price-per-reward optimality equations for this class of games. We also know that we can compare the values of gain and bias for different states. This will allow us to use a strategy improvement technique to find the optimal pair of state strategies. Subsequent strategies will be strictly better than previous ones (gain and bias will be used to compare strategies). A pair of strategies that can not be improved yields gain and bias that satisfy the average-price-per-reward optimality equations. Such a pair of strategies must exist because the sets of state strategies are finite, In the following we formalise this intuition.

We fix some finite average-price-per-reward game, Γ for the remainder of this section. Let μ^S and χ^S be state strategies for players Min and Max respectively, and let (G, B) be gain and bias functions such that $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma_{\mu^S \chi^S})$.

Remark 4.18. *The solution to the average-price-per-reward optimality equations $\text{Opt}_{\text{AvgPR}}(\Gamma_{\mu^S \chi^S})$ is not unique. However, in the remainder of this section, for technical convenience (proof of Theorem 4.19), by a solution we will mean a particular solution, i.e., a solution constructed as in the proof of Theorem 4.17. This solution has the following property, for every sun S , the bias of the vertex $\min(S)$ is equal to zero.* □

Given $s \in S^{\text{Min}}$ and $e = (s, s') \in E \setminus E_{\mu^S \chi^S}$, we say that e is an *improvement* of μ^S , with respect to χ^S , if:

1. $G(s) > G(s')$, or

2. $G(s) = G(s')$ and $B(s) > \text{Val}(\ominus_e(G(s))) + B(s')$.

A strategy μ'^S is an improvement of μ^S with respect to χ^S if for every state s , either $\mu^S(s) = \mu'^S(s)$, or $\mu'^S(s) = s'$ and (s, s') is an improvement of μ^S with respect to χ^S . An improvement is strict if $\mu^S \neq \mu'^S$. An improvement of χ^S is defined similarly.

We say that χ^S , a state strategy for player Max, is a *best response* to μ^S , a state strategy of player Min, if there are no possible improvements of χ^S with respect to μ^S .

To prove the existence of mutual best response strategies we apply Theorem 4.19 and the fact that the set of edge strategies is finite, to average-price-per-reward games, in which all the states belong to only one player.

Theorem 4.19. *Let μ^S be a state strategy of player Min, χ^S a best response strategy of player Max, and (G, B) gain and bias such that $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma_{\mu^S \chi^S})$. If μ'^S is an improvement of μ^S with respect to χ^S , χ'^S is a best response to μ'^S , and $(G', B') \models \text{Opt}_{\text{AvgPR}}(\Gamma_{\mu'^S \chi'^S})$, then the following holds:*

1. $G(s) \geq G'(s)$, for all $s \in S$, and $G(s) > G'(s)$ for some $s \in S$, or
2. $G(s) = G'(s)$ and $B(s) \geq B'(s)$, for all $s \in S$.

Moreover, if $\mu^S \neq \mu'^S$ then $(G, B) \neq (G', B')$.

Proof. Consider the game graph $\Gamma_{\mu^S \chi^S}$. For every edge $e = (s, s')$, either i) $G(s) > G(s')$, or ii) $G(s) = G(s')$ and $B(s) \geq \text{Val}(\ominus_e(G(s))) + B(s')$.

We start by proving point 1. Observe that, for every edge (s, s') in $\Gamma_{\mu^S \chi^S}$, we have $G(s) \geq G(s')$. This implies that for every edge (s, s') , if $G(s) > G(s')$, then s is a ray state. This observation allows us to use the same argument as in Lemma 4.12 to prove that $G(s) \geq G'(s)$, for every state s . In particular, if an edge (s, s') , in $\Gamma_{\mu^S \chi^S}$, is such that $G(s) > G(s')$, or $B(s) > \text{Val}(\ominus_e(G(s))) + B(s')$ and s is a rim state, then $G(s) > G'(s)$.

It remains to prove point (2). We know that $G(s) = G'(s)$ for all $s \in S$. Let s be a vertex, and let S be a sun in $\Gamma_{\mu^S \chi^S}$ such that, $s \in S$. If s_0, \dots, s_k is the path from s to $\min(S)$ then, for every (s_i, s_{i+1}) ,

$$B(s_i) \geq \text{Val}(\varnothing_{(s_i, s_{i+1})}(G(s))) + B(s_{i+1}).$$

From the proof of point (1), we can assume that the sun S existed in the graph $\Gamma_{\mu^S \chi^S}$, and hence $B'(\min(S)) = B(\min(S)) = 0$. Hence, if we sum up and simplify, the k inequalities, we get:

$$B(s_0) \geq \sum_{i=0}^{k-1} \text{Val}(\varnothing_{(s_i, s_{i+1})}(G(s))) + B(s_k) = B'(s_0),$$

as $s_k = \min(S)$. To complete the proof, notice that if $\mu^S \neq \mu'^S$, then $B(s) > \text{Val}(\varnothing_{(s_i, s_{i+1})}(G(s))) + B(s_{i+1})$, for some i , and hence $B(s) > B'(s)$. \square

Corollary 4.20. *A solution to optimality equations for finite average-price-per-reward games exist.*

Proof. The set of edge strategies for both players is finite. This, together with Theorem 4.19, guarantees the existence of mutual best response edge strategies. The rest follows from Theorem 4.17. \square

Corollary 4.21. *In finite definable price-per-reward games, ε -optimal strategies are computable, provided that, under the optimal state strategies, every edge game admits rational ε -optimal moves.*

Proof. Corollary 4.20 ensures that the solutions to the average-price-per-reward games exist, and by Remark 4.8 this solution is definable. From this, by Corollary 4.11, we obtain definability of ε -optimal strategies. The existence of rational ε -optimal moves allows us to apply Proposition 2.7, to obtain the postulated computability result. \square

Theorem 4.22. *Finite definable average-price-per-reward games are decidable.*

Proof. The set of equations $\text{Opt}_{\text{AvgPR}}(\Gamma)$ is finite, thus the set of solutions is a set of finite-dimensional vectors over the reals. Remark 4.8 ensures that (G, B) , such that $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$, is definable. This implies decidability (Proposition 2.5). \square

Chapter 5

Games on hybrid automata with strong resets

Hybrid automata with strong resets are a subclass of hybrid automata, with strong restriction on discrete transitions. The restriction is that upon each discrete transition all continuous variables are reset to a value from some predefined set. In this chapter we will present a slightly different definition of a hybrid automaton with strong resets that is better geared towards the quasi-concurrent game setting. We will also formalise the notion of a quasi-concurrent game on a hybrid automaton with strong resets, referred to as a hybrid game with strong resets (Section 5.1). However, the main goal of this chapter is to show that certain hybrid games with strong resets are determined and decidable. The games in question are average-price-per-reward games (see Sections 2.2.2 and 5.1.1) and reachability-price games (see Sections 2.2.2 and 5.1.2).

We start this chapter by formally introducing hybrid automata with strong resets and hybrid games that are defined using the quasi-concurrent approach (see Section 2.3.3). The automata considered in this chapter will be double-weighted, and we will argue that a hybrid game with strong resets can be viewed as a game on an infinite price-reward game graph (introduced in Chapter 4). As a consequence, the

standard notions such as: strategies, runs, game values, etc., will be introduced using the methodology adopted in Chapter 4. In particular, for the case of average-price-per-reward hybrid games, we will argue that Theorem 4.10 is applicable. We will also prove the reachability-price analogue of this theorem, Theorem 5.14, which states that there is an optimality equation characterisation of game values for reachability-price games on hybrid automata with strong resets.

We conclude this chapter by presenting the two main results, i.e., that average-price-per-reward games (Theorem 5.18) and reachability-price games (Theorem 5.19) on hybrid automata with strong resets are determined and decidable. These two results are obtained in two steps. First, in Section 5.2.1, we introduce a finite equivalence relation on the state space of the hybrid automaton, and then use it to construct a finite price-reward game graph. Sections 5.2.2–5.2.3 are committed to showing, that solutions to the optimality equations for the respective games on the finite price-reward game graphs can be used to obtain solutions for the optimality equations for the respective games on the hybrid automata. This establishes that hybrid games, considered in this chapter, can be reduced to their finite graph counterparts.

5.1 Games on hybrid automata with strong resets

We introduce hybrid automata with strong resets and define games on these automata, and show how infinite price-reward game graphs can be used to interpret the quasi-concurrent hybrid game with strong resets. After defining the standard notions such as strategies, runs, etc., we proceed to define the two games of interest. In Section 5.1.1 we introduce the average-price-per-reward games on hybrid automata with strong resets, and explain how to apply Theorem 4.10 to the hybrid setting. In Section 5.1.2 we introduce reachability-price games, and prove Theorem 5.14, i.e., that values of the reachability-price games on hybrid automata with

strong resets, if they exist, are characterised by optimality equations.

Our definition of a hybrid automaton with strong resets varies from that used in [LPS00; BBC06; BBC09] and described in Section 2.3.1, as we hide the continuous dynamics of the automaton into guard functions. This approach allows for cleaner and more succinct notation and exposition, without loss of generality [BBJ⁺08; JLR09; RLJ10]. Additionally, we generalise the notion of a control action. We no longer require it to be an edge connecting two control locations in the control graph. We simply require that the set of control actions is finite.

Throughout this chapter, we assume that the games considered are non-blocking, i.e., the hybrid automaton and the hybrid game are defined in such a way that at every stage of the game both players can make a move (see Definition 5.5).

Hybrid automata with strong resets. As explained in Section 2.3, a *hybrid automaton* is an extension of a finite automaton, obtained through the addition of *real-valued variables*. The state of a hybrid automaton has two components, the control location of the underlying finite automaton, known as the control graph, and the valuation of the real-valued variables. Similarly, there are two kinds of transitions: *discrete* changes of control locations and *continuous* changes to variable values. Discrete transitions are immediate and their availability is subject to the current control location and the valuation of the real-valued variables, whereas continuous changes span over time and are governed by a control location-specific flow function.

Definition 5.1 (Hybrid Automaton with Strong Resets). *A a doubly-weighted hybrid automaton with strong resets (HASR), or simply a hybrid automaton with strong resets, \mathcal{H} is given by $\langle X, L, A, G, R, \pi, \rho \rangle$ where,*

- X is the set of n real-valued continuous variables (for some fixed $n \in \mathbb{N}$),
- L is the finite set of control locations,

- A is the finite set of control actions,
- $G : A \rightarrow 2^{S \times T}$ is the control action guard function,
- $R : A \rightarrow 2^S$ is the control action reset function,
- $\pi : S \times (A \times T) \rightarrow \mathbb{R}$ is the price function,
- $\rho : S \times (A \times T) \rightarrow \mathbb{R}$ is the reward function,

where $S = L \times \mathbb{R}^n$ denotes the set of states (a state is a control location and the set of values for the variables in X), and $T = \mathbb{R}_{\geq 0}$ denotes the set of time delays.

We say that a HASR is definable if all its components are definable.

As mentioned earlier, the definition of the hybrid automaton, as presented in this chapter, differs from the one presented in Section 2.3. It was constructed under the assumption that we will be considering transition semantics of hybrid automata with strong resets, and that the set of admissible runs will be restricted to alternating sequences of continuous and discrete transitions (timed action semantics). The latter restriction comes from the fact that we will be considering games in the quasi-concurrent setting (see Section 2.3.3).

The semantics of a HASR, \mathcal{H} , will be given in terms of a transition system $\mathcal{T}_{\mathcal{H}}$ (as seen in Section 2.3.1). As was the case with the definition of the hybrid automaton with strong resets, the definition of the transition system differs from that seen in Section 2.3, as it reflects the assumptions on the admissible executions of the hybrid automaton.

Definition 5.2 (Timed Action with Strong Resets Semantics). *Given a hybrid automaton with strong resets \mathcal{H} , the doubly-weighted labelled transition system $\mathcal{T}_{\mathcal{H}}$ consists of:*

- the set of states $S \subseteq L \times \mathbb{R}^n$,

- the set of transition labels $\Lambda \subseteq \mathbf{A} \times \mathbf{T}$, and a transition relation, $\rightarrow \subseteq \mathbf{S} \times (\mathbf{A} \times \mathbf{T}) \times \mathbf{S}$. The label $\tau = (a, t) \in \Lambda$ is called a timed action, and the transition $s \xrightarrow{a}_t s'$ (or simply $s \xrightarrow{\tau} s'$) is admissible if $(s, t) \in \mathbf{G}(a)$ and $s' \in \mathbf{R}(a)$.
- the price and reward functions are defined as $\pi(s, \tau)$ and $\rho(s, \tau)$, for every transition $s \xrightarrow{\tau} s'$, respectively.

Note that for a transition $s \xrightarrow{a}_t s'$, s' depends only on a and not on s or t . This is the strong reset property. Also note that the price and reward of a transition are independent of the successor state.

An execution of a hybrid automaton is modelled by a run of its transition system (see Section 2.2.2 for the definitions relating transition systems).

Remark 5.3. *Definition 5.1 does not place any restrictions on the continuous components of the hybrid automaton. In particular there are no non-negativity nor continuity requirements regarding the price and reward functions. It will be required, however, that the hybrid automaton is such that the price and reward functions are bounded, and the hybrid game, as introduced in Definition 5.5, is n -reward divergent¹. This is required by Theorem 4.10. Choosing $\rho(s, \tau) > c > 0$, for all $(s, \tau) \in \mathbf{S} \times (\mathbf{A} \times \mathbf{T})$, where c is a constant, is a simple way of ensuring n -reward divergence. \square*

To conclude the introduction of hybrid automata, we will briefly explain how our definition differs from that widely used in the literature, as introduced in Section 2.3.

Remark 5.4. *Traditionally, the continuous changes to the real-valued variables are governed by the control location-specific flow and inv predicates. See Definition 2.35 for the traditional definition of a hybrid automaton with strong resets, and Definition 2.36 for the traditional definition of its transition semantics.*

¹The notion of n -reward divergence was introduced in the context of price-reward game graphs, but as we argue later, a hybrid game with strong resets can be viewed as a game on an infinite price-reward game graph.

We now show how the strong reset property and the timed action semantics enable us to hide the complex dynamics of the hybrid automaton in the guard set of a control action.

Consider a timed action (a, t) admissible from a state $s = (\ell, x)$, i.e., $(s, t) \in G(a)$ and $s' = (\ell', x') \in R(a)$, where $a = (\ell, \ell')$. In the classical definition, this would mean that the following two transitions are admissible: $s \xrightarrow{t} s''$ and $s'' \xrightarrow{a} s'$. Under the strong reset property, this gives:

- $(s, t) \in G(a)$ iff there exists a differentiable function $f : [0, t] \rightarrow \mathbb{R}^n$, with the first derivative $\dot{f} : (0, t) \rightarrow \mathbb{R}^n$, such that: (1) $f(0) = x$ and $f(t) = x''$, and (2) for all $0 \leq t' \leq t$ the predicates $\text{inv}(\ell)[X := f(t')]$ and $\text{flow}(\ell)[X, \dot{X} := f(t'), \dot{f}(t')]$ are true;
- $s' \in R(a)$.

If the strong reset property was not in place, we would not be able to define the guard and reset sets independently, and hence to hide the dynamics in the guard set. \square

Hybrid games with strong resets. Hybrid games with strong resets are played on hybrid automata with strong resets. The rules of the game reflect the intended application of the model. Player Min models the role of the control program, and player Max models the worst-case behaviour of the environment, i.e., the factors that are out of control. We are considering the quasi-concurrent game setting, i.e., the game is played in rounds, and each round consist of three steps that model the interaction of the controller with the environment. If a system is in some state, the control program chooses to execute a certain timed action, this constrains the possible behaviour, and the final transition is determined by the environment.

Definitions 5.1 and 5.2 emphasise the mechanics of the hybrid automaton. However, we will need a notation that is better geared towards the game setting; that puts more emphasis on actions available to players. For that purpose we define

the *move* function $M : S \rightarrow 2^{A \times T}$ as $M(s) = \{(a, t) : (s, t) \in G(a)\}$. Note that M is definable if G is definable.

Definition 5.5 (Hybrid Game with Strong Resets). *A hybrid game with strong resets (HGSR), or simply a hybrid game, $\Gamma = \langle \mathcal{H}, M^{\text{Min}}, M^{\text{Max}} \rangle$ consists of:*

- a HASR $\mathcal{H} = \langle X, L, A, G, R, \pi, \rho \rangle$,
- a Min-move function $M^{\text{Min}} : S \rightarrow 2^{A \times T}$,
- a Max-move function $M^{\text{Max}} : S \times (A \times T) \rightarrow 2^{A \times T}$, and
- a payoff function $P : \text{Runs} \rightarrow \mathbb{R}$.

We require that for all $s \in S$, we have $M^{\text{Min}}(s) \subseteq M(s)$, and that for all $\tau \in M^{\text{Min}}(s)$, we have $M^{\text{Max}}(s, \tau) \subseteq M(s)$. Without loss of generality, we assume that Γ is non-blocking, i.e., for all $s \in S$, we have $M^{\text{Min}}(s) \neq \emptyset$, and that for all $\tau \in M^{\text{Min}}(s)$, we have $M^{\text{Max}}(s, \tau) \neq \emptyset$. If \mathcal{H} and the move functions are definable, then we say that Γ is definable.

A quasi-concurrent hybrid game is played in rounds (as discussed in Section 2.3.3). In every round, the following three steps are performed by the two players Min and Max from the current state $s \in S$.

1. Player Min proposes a timed action $\tau \in M^{\text{Min}}(s)$.
2. Player Max responds by choosing a timed action $\tau' = (a', t') \in M^{\text{Max}}(s, \tau)$.
This choice determines the price and reward contribution of the round ($\pi(s, \tau')$ and $\rho(s, \tau')$, respectively).
3. Player Max chooses a state $s' \in R(a')$, i.e., such that $s \xrightarrow{\tau'} s'$. The state s' becomes the current state for the next round.

A *play* of the game Γ from state s is a (possibly infinite) sequence $\wp = s_0 \xrightarrow{\tau_1, \tau'_1} s_1 \xrightarrow{\tau_2, \tau'_2} s_2 \cdots$ such that $s_0 = s$, and for all $i \geq 0$, we have $\tau_{i+1} \in M^{\text{Min}}(s_i)$,

and $\tau'_{i+1} \in \mathbf{M}^{\text{Max}}(s_i, \tau_{i+1})$, and $s_{i+1} \in \mathbf{R}(a'_{i+1})$, where $\tau'_{i+1} = (a'_{i+1}, t'_{i+1})$. Note that if $\wp = s_0 \xrightarrow{\tau_1, \tau'_1} s_1 \xrightarrow{\tau_2, \tau'_2} s_2 \cdots$ is a play of the hybrid game Γ , then the sequence $\omega = s_0 \xrightarrow{\tau'_1} s_1 \xrightarrow{\tau'_2} s_2 \cdots$ is a run of the hybrid automaton \mathcal{H} .

Remark 5.6. *Our definition of the game round generalises the definition found in Section 2.3.3, where actions available to players are given through a partition of the set of control actions. We will show how the classical definition can be captured using the one presented in this chapter. Suppose that we have a partition of the set of control actions, given by $\mathbf{A} = \mathbf{A}^{\text{Min}} \cup \mathbf{A}^{\text{Max}}$. Given a state s , we would define the player Min's move function, $\mathbf{M}^{\text{Min}}(s)$, as $\{(a, t) : a \in \mathbf{A}^{\text{Min}} \text{ and } (s, t) \in \mathbf{G}(a)\}$ and the player Max's move function, $\mathbf{M}^{\text{Max}}(s, (a, t))$, as $\{(a', t') : a' \in \mathbf{A}^{\text{Max}} \text{ and } t' \leq t\} \cup \{(a, t)\}$, where $(a, t) \in \mathbf{M}^{\text{Min}}(s)$. \square*

We now define the standard notions such as strategies, game values, and determinacy. Although these definitions are similar to the ones introduced in Sections 2.2.2 and 4.1, they account for the quasi-concurrent game setting. We present them in full to assure the clarity of the presentation. Later in this section we will explain how strategies in a hybrid game relate to strategies on an infinite price-reward games graphs (see Section 4.1).

A strategy of player Min is a function $\mu : \text{Runs}_{\text{fin}} \rightarrow \mathbf{A} \times \mathbf{T}$ such that, $\mu(\omega) \in \mathbf{M}^{\text{Min}}(s)$, where ω is a finite run ending in s . We say that μ , a strategy of player Min, is positional if for every two runs $\omega, \omega' \in \text{Runs}_{\text{fin}}$ we have, if ω and ω' end in the same state then $\mu(\omega) = \mu(\omega')$, i.e., it can be viewed as a function $\mu : \mathbf{S} \rightarrow \mathbf{A} \times \mathbf{T}$. We will write Σ^{Min} for the set of all strategies of player Min, and Π^{Min} for the subset of all positional strategies.

A strategy of player Max is a function $\chi : (\text{Runs}_{\text{fin}} \times (\mathbf{A} \times \mathbf{T})) \rightarrow ((\mathbf{A} \times \mathbf{T}) \times \mathbf{S})$ such that if $\mu(\omega, \tau) = (\tau', s')$, then $\tau' \in \mathbf{M}^{\text{Min}}(s, \tau)$ and $s' \in \mathbf{R}(a)$, where ω is a finite run ending in s , and $\tau = (a, t)$. We say that χ , a strategy of player Max, is positional if for every two runs $\omega, \omega' \in \text{Runs}_{\text{fin}}$ we have, if ω and ω' end in the same state s , then $\chi(\omega, \tau) = \chi(\omega', \tau)$ for every $\tau \in \mathbf{M}^{\text{Min}}(s)$, i.e., it can be viewed as a

function $\chi : (S \times (A \times T)) \rightarrow ((A \times T) \times S)$. We will write Σ^{Max} for the set of all strategies of player Max, and Π^{Max} for the subset of all positional strategies.

Given $\mu \in \Sigma^{\text{Min}}$ and $\chi \in \Sigma^{\text{Max}}$, let $\wp = s_0 \xrightarrow{\tau_1, \tau'_1} s_1 \xrightarrow{\tau_2, \tau'_2} s_2 \dots$ be the unique play from the state s_0 . We will write $\text{Run}(s_0, \mu, \chi)$ to denote the unique run $\omega = s_0 \xrightarrow{\tau'_1} s_1 \xrightarrow{s'_2} \dots$ arising from the play \wp .

We can now introduce the concept of the value of a hybrid game from a state. We do this by defining a zero-sum game in strategic form, which is specific to the given state (recall Section 2.2.2).

Given a state s , let $\mathcal{D}_s = \langle \Sigma^{\text{Min}}, \Sigma^{\text{Max}}, P(\text{Run}(s, \cdot, \cdot)) \rangle$ be a zero-sum game in strategic form. We say that the game Γ is *determined* from s if \mathcal{D}_s is determined, i.e., $\text{Val}_*(\mathcal{D}_s) = \text{Val}^*(\mathcal{D}_s)$, and *positionally determined* if

$$\text{Val}(\mathcal{D}_s) = \inf_{\mu \in \Pi^{\text{Min}}} \text{Val}^\mu(\mathcal{D}_s) = \sup_{\chi \in \Pi^{\text{Max}}} \text{Val}_\chi(\mathcal{D}_s).$$

We say that the price-reward game Γ is determined if it is determined from every state.

For simplicity we will write $\text{Val}(s)$ rather than $\text{Val}(\mathcal{D}_s)$, in the context of price-reward games, so Val can be viewed as a partial function $S \rightarrow \mathbb{R}$.

We will say that a price-reward game Γ is *decidable* if the partial function $\text{Val} : S \rightarrow \mathbb{R}$ is decidable (recall Definition 2.17). We emphasise that Val is a partial function because Γ does not have to be determined from every state.

Infinite price-reward game graphs. A hybrid game with strong resets can be viewed as a game on an infinite price-reward game graph, with fixed costs and rewards assigned to edges (see Chapter 4). The graph consists of three groups of states: states of the hybrid automaton, states that represent the choice of player Min in the first step of the round, and states that represent the response to that choice of player Max in the second step of the round. The edges reflect the admissible choices, that the players can make at every step.

To distinguish the HGSR Γ from its corresponding price-reward game graph, we will use a prime, i.e., Γ' . We will write S' to denote the set of states of this game graph, and we have:

$$S' \subseteq S \cup \underbrace{(S \times (A \times T))}_{\text{choices of player Min}} \cup \underbrace{(A \times T)}_{\text{responses of player Max}}$$

The set S' is equal to the minimum set that contains S , and all the states reachable from S using the edge relation E' . The set E' is the minimum set such that:

Step 1 for every $s \in S$ and $\tau \in M^{\text{Min}}(s)$ there is an edge $(s, (s, \tau)) \in E'$,

Step 2 for every (s, τ) , and $\tau' \in M^{\text{Max}}(s, \tau)$ there is an edge $((s, \tau), \tau') \in E'$,

Step 3 for every $\tau' = (a', t')$, and $s' \in R(a')$ there is an edge $(\tau', s') \in E'$.

Now we can define $\Gamma' = \langle S, S' \setminus S, E', \pi', \rho' \rangle$. An edge $e = ((s, \tau), (a', t'))$ has price $\pi'(e) = \pi(s, t')$ and reward $\rho'(e) = \rho(s, t')$. These edges correspond to the price and reward determining aspect of the actual transitions of the hybrid automaton. The remaining edges have both the price and reward equal to zero; they only represent the steps of a game round.

Remark 5.7. For all $(a, t), (a', t') \in S'$, if $a = a'$ then the sets of states reachable from those two states, with respect to E' , are equal. This is a consequence of the strong reset property of \mathcal{H} . \square

Remark 5.8. It is clear that plays of Γ directly correspond to runs on Γ' . Moreover, any run of Γ' , starting in S , uniquely determines a play in Γ , and hence a run of \mathcal{H} . On the other hand, one should observe that in general strategies in Γ' are stronger than those in Γ , as they act on plays of Γ , rather than just runs of Γ — one can infer a run from a play, but the converse is not true. However, positional strategies in both Γ and Γ' are equally expressive. This is sufficient, as we will show that the hybrid games of interest are positionally determined. \square

Remark 5.8 will enable us to lift the concepts introduced for price-reward games to hybrid price-reward games. We will say that the hybrid game Γ has a property P if Γ' has this property.

5.1.1 Hybrid average-price-per-reward games

In the following, we lift the concept of average-price-per-reward games, as defined in Chapter 4, to hybrid games with strong resets.

As was the case with price-per-reward games on transition systems (see Section 2.2.2 and price-per-reward games on price-reward game graphs (see Chapter 4), we need to define both the upper and lower payoffs. The payoff of the game depends on the actual execution of the automaton, which arises from a play of a hybrid game.

We now recall the definition of the average-price-per-reward payoffs, as defined in Section 2.2.2. Note that, the transition system is defined as in Definition 5.2, as we have adopted the timed action semantics. The lower and upper payoffs are defined as follows. For a run $\omega = s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s_2 \cdots$ of \mathcal{H} , we define the lower payoff \mathcal{A}_* and the upper payoff \mathcal{A}^* as

$$\mathcal{A}_*(\omega) = \liminf_{n \rightarrow \infty} \frac{\sum_{i=0}^{n-1} \pi(s_i, \tau_{i+1})}{\sum_{i=0}^{n-1} \rho(s_i, \tau_{i+1})},$$

and

$$\mathcal{A}^*(\omega) = \limsup_{n \rightarrow \infty} \frac{\sum_{i=0}^{n-1} \pi(s_i, \tau_{i+1})}{\sum_{i=0}^{n-1} \rho(s_i, \tau_{i+1})}.$$

Note that the payoffs for the hybrid version of price-per-reward game are exactly the same as the average-price-per-reward payoffs for runs starting in $S \subseteq S'$ in Γ' .

Similarly to average-price-per-reward games on finite game graphs (see Section 2.2.2) and price-reward game graphs (see Section 4.1.1) we will provide an optimality-equation characterisation of game values.

We will also say that $\text{Opt}_{\text{AvgPR}}(\Gamma')$, the set of average-price-per-reward op-

tinality equations for the price-reward game Γ' induced by Γ , is the set of average-price-per-reward optimality equations for the hybrid game Γ , which is denoted by $\text{Opt}_{\text{AvgPR}}(\Gamma)$. Let $G, B : \mathbf{S} \cup (\mathbf{S} \times (\mathbf{A} \times \mathbf{T})) \cup (\mathbf{A} \times \mathbf{T}) \rightarrow \mathbb{R}$. The average-price-per-reward optimality equations for Γ' take the following form: if $s \in \mathbf{S}$, then

$$G(s) = \min\{G(s, \tau) : \tau \in \mathbf{M}^{\text{Min}}(s)\}, \quad (5.1)$$

$$B(s) = \inf\{B(s, \tau) : \tau \in \mathbf{M}^{\text{Min}}(s) \text{ and } G(s, \tau) = G(s)\}; \quad (5.2)$$

if $s \in \mathbf{S}$ and $\tau \in \mathbf{M}^{\text{Min}}(s)$, then

$$G(s, \tau) = \max\{G(a') : (a', t') \in \mathbf{M}^{\text{Max}}(s, \tau)\}, \quad (5.3)$$

$$B(s, \tau) = \sup\{\pi(s, a', t') - \rho(s, a', t') \cdot G(a') + B(a') : \\ (a', t') \in \mathbf{M}^{\text{Max}}(s, \tau) \text{ and } G(a') = G(s, \tau)\}; \quad (5.4)$$

and if $a \in \mathbf{A}$, then

$$G(a) = \max\{G(s) : s \in \mathbf{R}(a)\}, \quad (5.5)$$

$$B(a) = \sup\{B(s) : s \in \mathbf{R}(a) \text{ and } G(s) = G(a)\}. \quad (5.6)$$

The last pair of equations is a generic pair of equations for all states $(a, t) \in \mathbf{S}'$. This is valid by Remark 5.7. We have written the equations taking into account the fixed price and rewards in Γ' . In particular, edge games, which were used to introduce average-price-per-reward optimality equations in Section 4.1.2, do not appear in the above equations.

Remark 5.9. *As noted above, Remark 5.7 implies that, if $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$, then the value of gain and bias in a state $(a, t) \in \mathbf{S}'$ depends solely on $a \in \mathbf{A}$. For this reason, we will often abuse notation, and treat the gain and bias functions as functions of the following type: $G, B : \mathbf{S} \cup (\mathbf{S} \times (\mathbf{A} \times \mathbf{T})) \cup \mathbf{A} \rightarrow \mathbb{R}$. \square*

We conclude with the following remark, which establishes the average-price-per-reward optimality equation characterisation of game values.

Remark 5.10. *Requiring Γ to be $\Omega(n)$ -divergent and price convergent enables us to use the optimality equation characterisation, and Theorem 4.10 in particular, from Section 4.1.2. Using Remark 5.7 and the fact that \mathbf{A} is a finite set, we guarantee that gain has a finite range, and that bias is bounded. The fact that the range of gain is finite justifies the use of \min and \max , instead of \inf and \sup , in the equations 5.1, 5.3, and 5.5. \square*

Average-price games. In hybrid average-price games [BBJ⁺08], a variant of hybrid average-price-per-reward games, the price of a timed action is averaged over the total number of timed actions. In other words, it is an average-price-reward game in which the reward of a timed action is constant and equal to 1. The respective payoff functions look as follows:

$$\mathcal{A}_*(\omega) = \liminf_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} \pi(s_i, \tau_{i+1}),$$

and

$$\mathcal{A}^*(\omega) = \limsup_{n \rightarrow \infty} \frac{1}{n} \sum_{i=0}^{n-1} \pi(s_i, \tau_{i+1}).$$

As we will explain in Section 5.2.2, hybrid average-price games are simpler than hybrid average-price-per-reward games. This, however, makes them more suitable for example purposes.

Example 5.11. *We provide a simple example of a hybrid automaton that will be used, later in the chapter, to illustrate our techniques. The automaton consists of two continuous variables, a single control location and two control actions. We define the hybrid automaton $\mathcal{H} = \langle \mathbf{X}, \mathbf{L}, \mathbf{A}, \mathbf{G}, \mathbf{R}, \pi \rangle$ to consist of the following:*

- *the set of variables $\mathbf{X} = \{x, y\}$,*

- the set of control locations $\mathbf{L} = \{\ell\}$,
- the set of control actions $\mathbf{A} = \{a, b\}$,
- the control action guard function given by

$$\mathbf{G}(c) = \begin{cases} (V_3 \times I_3) \cup (\mathbf{S} \times I_1) & \text{if } c = a, \text{ and} \\ (V_3 \times I_2) & \text{if } c = b, \end{cases}$$

- the control action reset function given by $\mathbf{R}(c) = \mathbf{L} \times V_1$, where $c \in \mathbf{A}$,
- the price function given by $\pi(x, y, (c, t)) = -(t + x^2 + y^2)$, where $c \in \mathbf{A}$,
- the reward function, $\rho \equiv 1$,

where the set of states, \mathbf{S} , of \mathcal{H} is equal to $\{\ell\} \times \mathbb{R}^2$, and the sets used in the definition, V_1, V_2, V_3, I_1, I_2 , and I_3 , (see Fig 5.1 for the graphical interpretation of the components of \mathcal{H}) are defined as follows:

$$V_1 = \{(x, y) : x + y \geq 10\},$$

$$V_2 = \mathbb{R}^2 \setminus V_1,$$

$$V_3 = \{(x, y) : y^2 + x^2 \leq 0\},$$

$$I_1 = (1, 2),$$

$$I_2 = (3, 4), \text{ and}$$

$$I_3 = (5, 6).$$

We now define a hybrid game Γ on \mathcal{H} . First, recall that $\mathbf{M}(s) = \{(c, t) : (s, t) \in \mathbf{G}(c)\}$. Second, we define $\Gamma = \langle \mathcal{H}, \mathbf{M}^{\text{Min}}, \mathbf{M}^{\text{Max}}, \mathcal{A} \rangle$, an average-price game

(the reward is constant and equal to 1), by setting:

$$\mathbf{M}^{\text{Min}}(s) = \mathbf{M}(s) \cap (\{a\} \times \mathbb{T}) = \begin{cases} \{a\} \times (I_3 \cup I_1) & \text{if } s \in V_3, \\ \{a\} \times I_1 & \text{otherwise,} \end{cases}$$

and

$$\mathbf{M}^{\text{Max}}(s, (a, t)) = \begin{cases} \{(a, t)\} & \text{if } t \in I_1 \\ \{(a, t)\} \cup (\{b\} \times I_2) & \text{if } t \in I_3. \end{cases}$$

The payoff function, \mathcal{A} , is the average-price-per-reward payoff function, as introduced in this section.

An example game round of Γ looks as follows. Suppose that the current state of \mathcal{H} is $s = (\ell, (1, -2)) \in (\{\ell\} \times V_3)$, and that player Min proposes a timed action $\tau = (a, 5.5) \in (\{a\} \times I_3) \subseteq \mathbf{M}^{\text{Min}}(s)$. This choice, of player Min, allows player Max to choose between complying, i.e., executing τ , or choosing a different timed action from the set $(\{b\} \times I_2) \subseteq \mathbf{M}^{\text{Max}}(s, \tau)$. Suppose that he decides to override the choice of player Min by choosing $\tau' = (b, 3.5) \in (\{b\} \times I_2) \subseteq \mathbf{M}^{\text{Max}}(s)$. To complete the game round player Max must choose a new state $s' \in \mathbf{R}(b)$, say $s' = (\ell, (5, 5)) \in (\{\ell\} \times V_1) = \mathbf{R}(b)$. This game round resulted in a transition $(\ell, (1, -2)) \xrightarrow{b, 3.5} (\ell, (5, 5))$ and incurred a price of $\pi((\ell, (1, -2)), (b, 3.5)) = -(3.5 + 1^2 + (-2)^2) = -7.5$ and a reward of 1.

To conclude this example, we will present the average-price-per-reward optimality equations, $\text{Opt}_{\text{AvgPR}}(\Gamma)$, for Γ . The equations look as follows. For $s \in (\{\ell\} \times V_3)$ we have:

$$\begin{aligned} G(s) &= \min \left\{ G(s, (a, t)) : t \in I_1 \cup I_3 \right\} \\ B(s) &= \inf \left\{ B(s, (a, t)) : t \in (I_1 \cup I_3) \text{ and } G(s, (a, t)) = G(s) \right\}. \end{aligned}$$

For $s \in (\{\ell\} \times (\mathbf{S} \setminus V_3))$ we have:

$$\begin{aligned} G(s) &= \min \left\{ G(s, (a, t)) : t \in I_1 \right\} \\ B(s) &= \inf \left\{ B(s, (a, t)) : t \in I_1 \text{ and } G(s, (a, t)) = G(s) \right\}. \end{aligned}$$

For $s = (\ell, x, y) \in (\{\ell\} \times V_3)$ and $\tau = (a, t)$, with $t \in I_3$, we have:

$$\begin{aligned} G(s, \tau) &= \min\{G(a), G(b)\} \\ B(s, \tau) &= \begin{cases} \max \left\{ -(t + x^2 + y^2) - G(a) + B(a), \right. \\ \quad \left. \sup_{t' \in I_2} -(t' + x^2 + y^2) - G(b) + B(b) \right\} & \text{if } G(a) = G(b), \\ \sup_{t' \in I_2} -(t' + x^2 + y^2) - G(b) + B(b) & \text{if } G(b) = G(s, \tau), \\ -(t + x^2 + y^2) - G(a) + B(a) & \text{if } G(a) = G(s, \tau). \end{cases} \end{aligned}$$

For $s = (\ell, x, y) \in \mathbf{S}$, and $t \in I_1$ we have:

$$\begin{aligned} G(s, (a, t)) &= G(a) \\ B(s, (a, t)) &= -(t + x^2 + y^2) - G(a) + B(a). \end{aligned}$$

For $c \in \mathbf{A} = \{a, b\}$ we have:

$$\begin{aligned} G(c) &= \sup\{G(s) : s \in \{\ell\} \times V_1\} \\ B(c) &= \sup\{B(s) : s \in (\{\ell\} \times V_1) \text{ and } G(a) = G(s)\}. \end{aligned}$$

In Section 5.2.2 we will show how to compute the solutions to the average-price-per-reward optimality equations $\text{Opt}(\Gamma)$. □

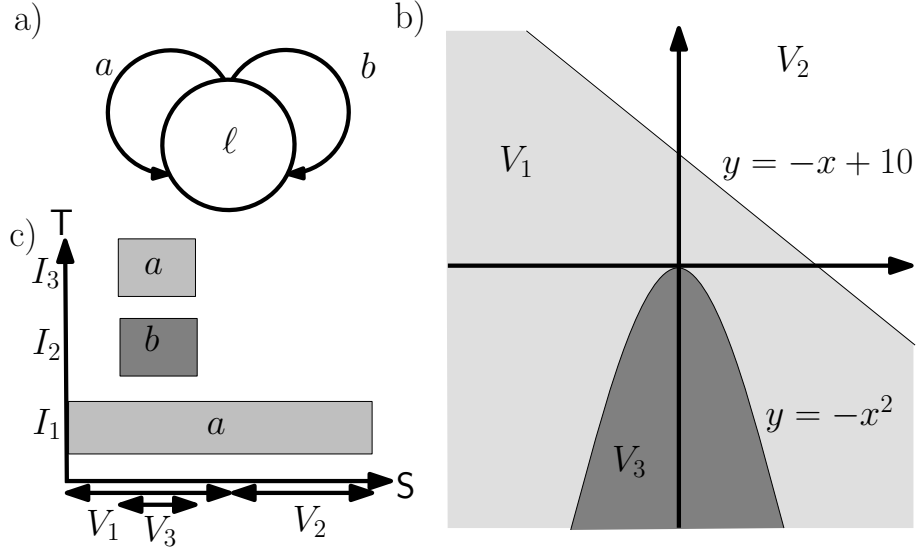


Figure 5.1: The hybrid automaton \mathcal{H} discussed in Example 5.11. a) Graph structure underlying \mathcal{H} . b) State space of \mathcal{H} . c) Guard function of \mathcal{H} .

5.1.2 Hybrid reachability-price games

In the following, we lift the concept of reachability-price-per-reward games, as defined in Section 2.2.2, to hybrid games with strong resets. The key result of this section is Theorem 5.14, which establishes an optimality equation characterisation of game values in a hybrid reachability-price game. We will use it to prove determinacy of hybrid reachability-price games, by showing that solutions to the reachability-price optimality equations exist (see Section 5.2.3).

As was the case with reachability-price games on transition systems (see Section 2.2.2) we need to define a single payoff function. The payoff of the game depends on the actual execution of the automaton, which arises from a play of a hybrid game.

A *hybrid reachability-price game with strong resets* (Γ, F) consists of a hybrid game with strong resets Γ and of a (definable) set $F \subseteq S$ of *final* states.

We now recall the definition of the reachability-price payoff, as defined in Section 2.2.2. Note that, the transition system is defined as in Definition 5.2, as we

have adopted the timed action semantics. For a run $\omega = s_0 \xrightarrow{\tau_1} s_1 \xrightarrow{\tau_2} s_2 \cdots$ of \mathcal{H} , we define

$$\text{Stop}(\omega) = \inf\{n : s_n \in \mathbf{F}\}.$$

The reachability-price payoff, \mathcal{P} is defined as

$$\mathcal{P}(\omega) = \begin{cases} \sum_{i=0}^{\text{Stop}(\omega)-1} \pi(s_i, \tau_{i+1}) & \text{if } \text{Stop}(\omega) < \infty, \\ \infty & \text{otherwise.} \end{cases}$$

As in the case of finite reachability-price games (see Section 2.2.2), we will now provide an optimality equation characterisation of game values. Let $P : \mathbf{S} \cup (\mathbf{S} \times (\mathbf{A} \times \mathbf{T})) \cup \mathbf{A} \rightarrow \mathbb{R}$ and let $D : \mathbf{S} \cup (\mathbf{S} \times (\mathbf{A} \times \mathbf{T})) \cup \mathbf{A} \rightarrow \mathbb{N}$. We will call the functions *price* and *distance*, respectively.

We say that a pair of price and distance functions, (P, D) , is a solution of *reachability-price optimality equations*, denoted by $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma, \mathbf{F})$, if the following conditions hold for all states $s \in \mathbf{S}$. If $s \in \mathbf{F}$ then $P(s) = D(s) = 0$. If $s \notin \mathbf{F}$ then we have:

$$P(s) = \inf \left\{ P(s, \tau) : \tau \in \mathbf{M}^{\text{Min}}(s) \right\}, \quad (5.7)$$

$$D(s) = \min \left\{ 1 + d : \right. \\ \left. P(s) = \inf \left\{ P(s, \tau) : D(s, \tau) = d \text{ and } \tau \in \mathbf{M}^{\text{Min}}(s) \right\} \right\}; \quad (5.8)$$

if $\tau \in \mathbf{M}^{\text{Min}}(s)$, we have:

$$P(s, \tau) = \sup \left\{ \pi(s, (a', t')) + P(a') : (a', t') \in \mathbf{M}^{\text{Max}}(s, \tau) \right\}, \quad (5.9)$$

$$D(s, \tau) = \max \left\{ 1 + d : \right. \\ \left. P(s, \tau) = \sup \left\{ \pi(s, (a', t')) + P(a') : \right. \right. \\ \left. \left. D(a') = d \text{ and } (a', t') \in \mathbf{M}^{\text{Max}}(s, \tau) \right\} \right\}; \quad (5.10)$$

and if $\tau' = (a', t') \in M^{\text{Max}}(s, \tau)$, we have:

$$\begin{aligned} P(a') &= \sup \left\{ P(s') : s' \in R(a') \right\}, \\ D(a') &= \max_{d' \in \mathbb{N}} \left\{ 1 + d' : P(a') = \sup \left\{ P(s') : D(s') = d' \text{ and } s' \in R(a') \right\} \right\}. \end{aligned}$$

Remark 5.12. *If Γ is definable then $\text{Opt}_{\text{Reach}}(\Gamma)$ is first-order expressible in \mathcal{M} . \square*

Remark 5.13. *Notice that the optimality equations for reachability-price games on hybrid automata refer directly to the hybrid game Γ , without resorting to the induced price-reward game graph Γ' , as was the case with average-price-per-reward games on hybrid automata. There, it was convenient since it enabled us to use the results from Chapter 4, whereas here these results are not applicable. \square*

As we have explained in Section 2.2.2, we can not apply the optimality equation characterisation to all the states of the hybrid automaton because not all states admit a finite value. First, we need to introduce two sets, $W^{\text{Max}} \subseteq S$, the set all of states from which player Max can prevent reaching F , and $W^{\text{Min}} \subseteq S \setminus W^{\text{Max}}$, the set all of states from which player Min can assure reaching F and from which the value of an average-price game is negative. Second, as argued in Section 2.2.2, for all states $s \in W^{\text{Max}}$ we have $\text{Val}(s) = \infty$, and for all states $s \in W^{\text{Min}}$ we have $\text{Val}(s) = -\infty$.

Let $S^{\text{fin}} = S \setminus (W^{\text{Max}} \cup W^{\text{Min}})$, and let Γ^{fin} be obtained from Γ by restricting the state space² to S^{fin} . The key result, which establishes that reachability-price optimality equations characterise the values of the hybrid reachability-price game, is as follows.

Theorem 5.14. *If $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}}, F)$ then for every state $s \in S^{\text{fin}}$, the hybrid reachability-price game (Γ^{fin}, F) from state s is determined and we have $\text{Val}(s) = P(s)$. Moreover, for every $\varepsilon > 0$, positional ε -optimal strategies exist*

²Note that in this case the superscript fin means, like in Section 2.2.2, that the value of the game exists, rather than that the set of states is finite — it does not have to be.

for both players.

Proof. The proof of the theorem is similar to that of Theorem 4.10, and follows from Lemmas 5.16 and 5.17. The lemmas imply that for all states s , we have $\text{Val}^*(s) \leq P(s)$ and $\text{Val}_*(s) \geq P(s)$. This in turn implies that $\text{Val}^*(s) = \text{Val}_*(s) = P(s)$. Moreover, the proofs of the lemmas are in fact constructive proofs of the existence of ε -optimal strategies. \square

Corollary 5.15. *If there exists (P, D) such that $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}}, F)$ and Γ^{fin} is definable, then positional ε -optimal strategies are definable.*

Proof. Positional strategies, as constructed in the proof of Lemmas 5.16–5.17, are definable if (P, D) , the solution to $\text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}}, F)$, is definable. \square

Lemma 5.16. *If $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}}, F)$, then for all $\varepsilon > 0$, there is $\mu_\varepsilon \in \Pi^{\text{Min}}$, such that for all $\chi \in \Sigma^{\text{Max}}$ and for all $s \in S^{\text{fin}}$, we have $\mathcal{P}(\text{Run}(s, \mu_\varepsilon, \chi)) \leq P(s) + \varepsilon$.*

Lemma 5.17. *If $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}}, F)$, then for all $\varepsilon > 0$, there is $\chi_\varepsilon \in \Pi^{\text{Max}}$, such that for all $\mu \in \Sigma^{\text{Min}}$ and for all $s \in S^{\text{fin}}$, we have $\mathcal{P}(\text{Run}(s, \mu, \chi_\varepsilon)) \geq P(s) - \varepsilon$.*

We omit the proof of Lemma 5.17 as it is similar to the proof of Lemma 5.16.

Proof of Lemma 5.16. We argue that a solution to reachability-price optimality equations can be seen as a witness for existence of positional ε -optimal strategies — we show how to use a solution of the reachability-price optimality equations to construct an ε -optimal strategy for player Min. The construction is fairly technical and can be seen to be focused on choosing locally optimal moves.

Before we start the actual proof, we formalise the notion of local optimality by introducing the concept of an ε' -optimal timed action. This concept will be used to construct the ε -optimal strategy, μ_ε for player Min. We write ε' to distinguish it from ε . Given an $\varepsilon' > 0$ and a state $s \in S^{\text{fin}}$ we will say that a timed action

$\tau \in \mathbf{M}^{\text{Min}}(s)$ is ε' -optimal, if the following hold:

$$\begin{aligned} P(s) &\geq P(s, \tau) - \varepsilon', \\ D(s) &= 1 + D(s, \tau). \end{aligned}$$

We will now construct a positional strategy μ_ε for player Min and then show that it is ε -optimal. For every state s , we choose $\tau = \mu_\varepsilon(s)$ to be a timed action that is $\frac{\varepsilon}{2^{D(s)+1}}$ -optimal, i.e.,

$$\begin{aligned} P(s) &\geq P(s', \tau) - \frac{\varepsilon}{2^{D(s)+1}} \\ D(s) &= 1 + D(s, \tau). \end{aligned}$$

The fact that there exists a solution to the reachability-price optimality equations, $\text{Opt}_{\text{Reach}}(\Gamma^{\text{fin}})$, ensures that such a timed action exists.

We now proceed to prove, that μ_ε defined in this way is indeed ε -optimal for player Min. For that purpose, let us fix a state s and some arbitrarily chosen strategy of player Max, $\chi \in \Sigma^{\text{Max}}$. Let $s_i \xrightarrow{\tau_{i+1}} s_{i+1}$ be the $(i+1)$ -th move of the $\text{Run}(s, \mu_\varepsilon, \chi)$, where $\tau = (a_{i+1}, t_{i+1}) \in \mathbf{M}^{\text{Max}}(s_i, \mu_\varepsilon(s_i))$. From the definition of μ_ε , regardless of χ , we have:

$$\begin{aligned} P(s_i) &\geq P(s_i, \mu_\varepsilon(s_i)) - \frac{\varepsilon}{2^{D(s_i)+1}}, \\ P(s_i, \mu_\varepsilon(s_i)) &\geq \pi(s_i, (a_{i+1}, t_{i+1})) + P(a_{i+1}), \\ P(a_{i+1}) &\geq P(s_{i+1}), \end{aligned}$$

and

$$\begin{aligned}
D(s_i) &\geq 1 + D(s_i, \mu_\varepsilon(s_i)), \\
D(s_i, \mu_\varepsilon(s_i)) &\geq 1 + D(a), \\
D(a) &\geq 1 + D(s_{i+1}).
\end{aligned}$$

As a consequence, because $D(s)$ was finite, there exists a $K \in \mathbb{N}$ such that $D(s_K) = P(s_K) = 0$, i.e., $s_K \in F$. If we sum up the $3 \cdot K$ inequalities for P we get:

$$\begin{aligned}
\sum_{i=0}^{K-1} P(s_i) + \sum_{i=0}^{K-1} P(s_i, \mu_\varepsilon(s_i)) + \sum_{i=0}^{K-1} P(a_{i+1}) &\geq \\
\sum_{i=0}^{K-1} P(s_i, \mu_\varepsilon(s_i)) - \sum_{i=0}^{K-1} \frac{\varepsilon}{2^{D(s_i)+1}} + \sum_{i=0}^{K-1} \pi(s_i, (a_{i+1}, t_{i+1})) + & \\
\sum_{i=0}^{K-1} P(a_{i+1}) + \sum_{i=0}^{K-1} P(s_{i+1}). &
\end{aligned}$$

The summations $\sum_{i=0}^{K-1} P(s_i, \mu_\varepsilon(s_i))$ and $\sum_{i=0}^{K-1} \pi(s_i, (a_{i+1}, t_{i+1}))$ occur on both sides on the inequality; summations $\sum_{i=0}^{K-1} P(s_i)$ and $\sum_{i=0}^{K-1} P(s_{i+1})$ differ by single term. This allows the inequality above to be simplified to:

$$\begin{aligned}
P(s_0) &\geq -\varepsilon \sum_{i=0}^{K-1} \frac{1}{2^{D(s_i)+1}} + \sum_{i=0}^{K-1} \pi(s_i, (a_{i+1}, t_{i+1})) + P(s_K) \geq \\
&-\varepsilon \sum_{i=0}^{\infty} \frac{1}{2^{i+1}} + \mathcal{P}(\text{Run}(s, \mu_\varepsilon, \chi)).
\end{aligned}$$

To obtain the second inequality, we apply the definition of the reachability-price payoff \mathcal{P} and the fact that $P(s_K) = 0$. Additionally observe that, the value of the infinite summation, in the term adjacent to ε , is 1, and hence we obtain the following:

$$P(s) \geq \mathcal{P}(\text{Run}(s, \mu_\varepsilon, \chi)) - \varepsilon.$$

This, due to the arbitrary choice of s and χ , yields the desired result. \square

5.2 Solving hybrid games with strong resets

So far we have introduced hybrid games with strong resets, games on hybrid automata with strong resets, in general, and discussed the hybrid of versions average-price-per-reward and reachability-price games in detail. The two main results of this section are as follows:

Theorem 5.18. *Average-price-per-reward games on hybrid automata with strong resets are positionally determined and admit ε -optimal positional strategies.*

Theorem 5.19. *Reachability-price games on hybrid automata with strong resets are positionally determined and admit ε -optimal positional strategies.*

The proof of these theorems follows from optimality equation characterisations of game values, as presented in Sections 5.1.1 and 5.1.2, respectively, as well as from the results presented in this section, which establish that solutions to the optimality equations indeed exist.

This section is organised as follows. First, in Section 5.2.1, we introduce a finite equivalence relation \sim and explain how to use it to construct a special finite price-reward game graph. Second, in Section 5.2.2, we show that a solution to the optimality equations for the average-price-per-reward game on this special graph correspond to the solutions to the optimality equations for the average-price-per-reward game on a hybrid automaton with strong resets. Third, in Section 5.2.3, we establish a similar correspondence of optimality equations solutions for reachability-price games.

5.2.1 A finite abstraction

We now introduce a finitary equivalence relation over the state space of the hybrid game Γ . It is used to construct a finite price-reward game graph $\hat{\Gamma}$. We later show that, solutions to $\text{Opt}_{\text{AvgPR}}(\hat{\Gamma})$ and to $\text{Opt}_{\text{Reach}}(\hat{\Gamma}, \hat{F})$ coincide with the solu-

tions to $\text{Opt}_{\text{AvgPR}}(\Gamma)$ and to $\text{Opt}_{\text{Reach}}(\Gamma, F)$ respectively (Theorems 5.26 and 5.31 respectively).

In the hybrid game Γ , each step of a round has a hybrid nature, i.e., consists of both a discrete and a continuous component. In the first two steps, players Min and Max interact to make a discrete choice of a control and a continuous choice of time. The last step consists only of a continuous choice of a member of the reset set. Our aim is to separate these two aspects of a game round. This will enable us to construct a finite price-reward game graph whose structure will reflect the possible discrete choices, whereas the continuous choices will be modelled using edge games.

Finite equivalence. We start by introducing some auxiliary notation, so that it is easier to talk separately about the discrete and continuous behaviours of players. For $s \in S$ and $(a, t) \in M^{\text{Min}}(s)$, we define

$$A^{\text{Max}}(s, (a, t)) = \{a' \in A : (a', t') \in M^{\text{Max}}(s, (a, t)) \text{ for some } t' \in T\},$$

i.e., $A^{\text{Max}}(s, (a, t))$ is the set of control actions $a' \in A$, such that there is a valid response $(a', t') \in A \times T$ of player Max to the proposal (a, t) of player Min. For $s \in S$ and $t \in T$, let

$$A^{\text{MinMax}}(s, t) = \{(a, A^{\text{Max}}(s, (a, t))) : (a, t) \in M^{\text{Min}}(s)\},$$

i.e., the set $A^{\text{MinMax}}(s, t)$ is the set of all pairs $(a, A) \in A \times 2^A$, such that player Min can propose the timed action (a, t) from state s , and the set of control actions, appearing in valid responses of player Max to the proposal (a, t) of player Min, is exactly A .

Our intention is to treat two states as equivalent if they admit the same discrete steps in a game round. We will also need to discriminate between states that belong to different reset regions, despite admitting the same discrete steps. In

the case of reachability-price games, we will also have to account for the membership in the set of goal states, F .

Definition 5.20 (Finite Equivalence). *Let $\mathcal{R} = \{R_1, R_2, \dots, R_n\}$ be such that $R_i \subseteq S$ for all i . States s, s' are said to be equivalent, denoted by $s \sim_{\mathcal{R}} s'$, iff*

- $A^{\text{MinMax}}(s, T) = A^{\text{MinMax}}(s', T)$, and
- $s \in R_i$ iff $s' \in R_i$ for all $i \in \{1, \dots, n\}$.

We will use $\mathcal{R} = \{R(a) : a \in A\}$ for average-price-per-reward games, and $\mathcal{R} = \{R(a) : a \in A\} \cup \{F\}$ for reachability-price games. If the set \mathcal{R} is understood from the context, or if for the purpose of our discussion the exact identity of the set \mathcal{R} is not important then, we often write simply \sim instead of $\sim_{\mathcal{R}}$.

Remark 5.21. *Note that the first condition in the Definition 5.20 states that the functions $A^{\text{MinMax}}(s, \cdot), A^{\text{MinMax}}(s', \cdot) : T \rightarrow A \times 2^A$ have the same ranges. Therefore, if $Q \in S/\sim$, then it makes sense to set $A^{\text{MinMax}}(Q, T)$ to be the range of the function $A^{\text{MinMax}}(s, \cdot)$ for any $s \in Q$. \square*

Remark 5.22. *Observe that \sim is an equivalence relation on the set of states S , and that there are finitely many equivalence classes of \sim . Moreover, if Γ is definable then every equivalence class is also definable. \square*

From Γ to the finite game. The construction of $\hat{\Gamma}$ is built upon an idea to separate the discrete and continuous choices of both players. This separation is achieved by reconstructing the round of a game in such a way that first players make their discrete choices (in three steps) and then they make their continuous choices, which must be sound with respect to the discrete choices made earlier.

In $\hat{\Gamma}$, the discrete steps (1-3) of the reconstructed round are encoded by the choices of edges. The continuous choices (3a-3c) are encoded in the controllable and uncontrollable inputs, that are associated with the edges. Fix \mathcal{R} , as explained in

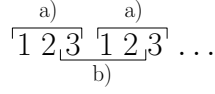


Figure 5.2: Numbers 1, 2, and 3 represent the three steps of a game round in a hybrid game with strong resets. a) the order in which the steps are carried out in the game Γ . b) the order in which the steps are carried out in the game $\hat{\Gamma}$.

this section. Let $a'' \in \mathbf{A}$ be the *current* control action — a state in the finite game graph $\hat{\Gamma}$. The players start with the following discrete choices

1. Max chooses $Q \in \mathbf{S}/\sim_{\mathcal{R}}$ such that $Q \subseteq \mathbf{R}(a'')$.
2. Min chooses a pair $(a, A) \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})$.
3. Max chooses a control action $a' \in A$, which becomes the current control action, and the players make their continuous choices:
 - (a) Max chooses $s \in Q$.
 - (b) Min chooses $t \in \mathbf{T}$ such that $(a, t) \in \mathbf{M}^{\text{Min}}(s)$ and $\mathbf{A}^{\text{Max}}(s, (a, t)) = A$.
 - (c) Max makes a choice of $t' \in \mathbf{T}$ such that $(a', t') \in \mathbf{M}^{\text{Max}}(s, (a, t))$.

The choices made by the players in steps 3, 1, and 2 of the hybrid game Γ are mapped to steps 1 and (a), 2 and (b), and 3 and (c) of the finite game $\hat{\Gamma}$. Additionally, the order of steps in the finite game is shifted with respect to the hybrid game (see Figure 5.2). Later in the chapter, we will prove that this shift in the game round order is valid (see Propositions 5.29 and 5.34).

The above finitary abstraction of choices made by players in every round of the hybrid game Γ is formalised by the following finite price-reward game graph $\hat{\Gamma} = (\hat{\mathbf{S}}, \hat{\mathbf{E}}, \hat{\mathcal{I}}, \hat{\mathcal{C}}, \hat{\mathcal{U}}, \hat{\pi}, \hat{\rho})$. The finite graph $(\hat{\mathbf{S}}, \hat{\mathbf{E}})$ is given by:

$$\begin{aligned} \hat{\mathbf{S}} &= \mathbf{A} \cup \mathbf{S}/\sim_{\mathcal{R}} \cup \{(Q, a, A) : Q \in \mathbf{S}/\sim \text{ and } (a, A) \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})\}, \\ \hat{\mathbf{E}} &= \{(a, Q) : Q \subseteq \mathbf{R}(a)\} \cup \\ &\quad \{(Q, (Q, a, A)) : (a, A) \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})\} \cup \{((Q, a, A), a') : a' \in A\}. \end{aligned}$$

In $\widehat{\Gamma}$, as in the infinite price-reward game graph Γ' , only one type of edge incurs a price and reward. The set of inputs for these edges reflects the possible continuous choices made by players during a round of the game. For the sake of simplicity, we will omit the specification of the input sets for the remaining edges; their price and reward is fixed, and equal to zero. Player Min's continuous choice consists of choosing the proposed time for every state. On the other hand, player Max's choice is slightly more complex. It consist of a choice of time, for every Min's choice, and a choice of the reset state. Formally the set of inputs is

$$\widehat{\mathcal{I}} = \underbrace{[\mathbf{S} \rightarrow \mathbf{A} \times \mathbf{T}]}_{\text{Min's inputs}} \cup \underbrace{\mathbf{S} \times [\mathbf{A} \times \mathbf{T} \rightarrow \mathbf{A} \times \mathbf{T}]}_{\text{Max's inputs}}.$$

Edges of the form $e = ((Q, a, A), a')$ will be the ones that incur the price and reward. These edges represent the discrete component of the actual transition of the original hybrid automaton. The remaining edges will bear a fixed price and reward equal to zero, making inputs obsolete.

Let $e = ((Q, a, A), a') \in \widehat{\mathbf{E}}$. The set of inputs for player Min is

$$\widehat{\mathcal{C}}(e) \subseteq [Q \rightarrow \mathbf{A} \times \mathbf{T}] \text{ (recall that } Q \subseteq S \text{)}$$

such that for every $s \in Q$ and $f \in \widehat{\mathcal{C}}(e)$, we have that

$$f(s) \in \mathbf{M}^{\text{Min}}(s) \text{ and } \mathbf{A}^{\text{Max}}(s, f(s)) = A,$$

i.e., the continuous choice $f(s)$, of player Min, is consistent with the agreed discrete choices. The set of inputs for player Max is

$$\widehat{\mathcal{U}}(e) \subseteq Q \times [\mathbf{A} \times \mathbf{T} \rightarrow \mathbf{A} \times \mathbf{T}]$$

such that for every $(s, f) \in \widehat{\mathcal{U}}(e)$, and $\tau \in \mathbf{M}^{\text{Min}}(s)$, consistent with the discrete

choices, we have that

$$s \in Q, f(\tau) \in M^{\text{Max}}(s, \tau) \text{ and } f(\tau) \in \{a'\} \times T,$$

i.e., the continuous choice (s, f) , of player Max, is consistent with the agreed choices.

Let $f \in \widehat{\mathcal{C}}(e)$ be the input chosen by player Min and $(s, f') \in \widehat{\mathcal{U}}(e)$ the input chosen by player Max. The price and reward of the edge e are then

$$\begin{aligned} \widehat{\pi}(e)(f, (s, f')) &= \pi(s, f'(f(s))), \text{ and} \\ \widehat{\rho}(e)(f, (s, f')) &= \rho(s, f'(f(s))). \end{aligned}$$

For the remaining edges we set $\widehat{\mathcal{C}}$ and $\widehat{\mathcal{U}}$ equal to \emptyset ; their price (reward) is fixed and equal to 0.

We define the finite price-reward game graph

$$\widehat{\Gamma} = \left(\widehat{\mathcal{H}}, \widehat{\mathcal{S}}^{\text{Min}}, \widehat{\mathcal{S}}^{\text{Max}}, \widehat{\mathcal{I}}, \widehat{\Theta}^{\text{Min}}, \widehat{\Theta}^{\text{Max}} \right),$$

where $(\widehat{\mathcal{S}}^{\text{Min}}, \widehat{\mathcal{S}}^{\text{Max}})$ is a partition of $\widehat{\mathcal{S}}$, given by $\widehat{\mathcal{S}}^{\text{Min}} = \mathcal{S}/\sim_{\mathcal{R}}$ and $\widehat{\mathcal{S}}^{\text{Max}} = \widehat{\mathcal{S}} \setminus \widehat{\mathcal{S}}^{\text{Min}}$.

Theorem 5.23. *If the hybrid price-reward game graph Γ is definable then the finite price-reward game graph $\widehat{\Gamma}$ is also definable.*

Proof. If the hybrid price-reward game graph Γ is definable we have the following:

- each of the finitely many equivalence classes of \sim is definable,
- the sets of inputs and player inputs in $\widehat{\Gamma}$ are definable,
- price and reward functions in $\widehat{\Gamma}$ are definable.

This renders $\widehat{\Gamma}$ definable. □

Remark 5.24 (Reachability-price games). *A reachability-price game on a HASR is given as (Γ, F) , and in this case the set \mathcal{R} , from the definition of $\sim_{\mathcal{R}}$, accounts*

for the goal states. The set of goal states \hat{F} , for the finite reachability-game $(\hat{\Gamma}, \hat{F})$, consists of those equivalence classes that are contained in F . If F was definable, then so is $(\hat{\Gamma}, \hat{F})$. \square

The following example should establish the intuition behind the construction of $\hat{\Gamma}$.

Example 5.25. Recall the hybrid automaton, \mathcal{H} , introduced in Example 5.11, and the respective average-price game Γ . We will show how to construct the finite game graph $\hat{\Gamma}$.

We start by constructing the equivalence $\sim_{\mathcal{R}}$, where $\mathcal{R} = \{R(a), R(b)\}$, as required for average-price-per-reward games.

The reset function, R , is defined as constant, over the set of control actions A , and its value is $L \times V_1$. As a consequence, the requirement that two equivalent states belong to the same elements of \mathcal{R} , introduces at least two equivalence classes: $L \times V_1$ and $L \times V_2$. On the other hand, for all states $s \in (S \setminus (L \times V_3))$ we have that $A^{\text{MinMax}}(s, T) = \{(a, \{a\})\}$, and for all states $s \in (L \times V_3)$ we have that $A^{\text{MinMax}}(s, T) = \{(a, \{a\}), (a, \{a, b\})\}$. The set V_1 is partitioned into two equivalence classes: $Q_1 = L \times V_3$ and $Q_2 = L \times (V_1 \setminus V_3)$ — it is required that all states within a single equivalence class admit the same $A^{\text{MinMax}}(\cdot, T)$ set. The remaining equivalence class is $Q_3 = L \times V_2$.

The finite priced game graph $\hat{\Gamma}$ obtained from Γ using $\sim_{\mathcal{R}}$ is depicted on Fig 5.3. Notice that the prices are fixed, and that we did not specify the input sets. We have done this because the reward function is constant and equal to 1, i.e., we are considering an average-price game for which the input mechanism is not necessary (see Section 5.2.2 and Example 5.30).

The fixed price of an edge $((Q, a, A), b)$, where $Q \in S / \sim$, and $(a, A) \in$

$A^{\text{MinMax}}(Q, \mathsf{T})$, and $b \in A$ is equal to the value of the following expression:

$$\begin{aligned}\widehat{\pi}((Q, a, A), b) &= \sup_{s \in Q} \inf_{t \in \mathsf{T}_{s, (a, A)}^{\text{Min}}} \sup_{t' \in \mathsf{T}_{s, (a, t), b}^{\text{Max}}} \pi(s, (b, t')), \text{ where} \\ \mathsf{T}_{s, (a, A)}^{\text{Min}} &= \{t \in \mathsf{T} : (a, t) \in \mathsf{M}^{\text{Min}}(s) \text{ and } A = \mathsf{A}^{\text{Max}}(s, (a, t))\}, \\ \mathsf{T}_{s, (a, t), b}^{\text{Max}} &= \{t' \in \mathsf{T} : (b, t') \in \mathsf{M}^{\text{Max}}(s, (a, t))\}.\end{aligned}$$

We will explain, however, how to construct the input sets, to help establish the intuition behind the construction of $\widehat{\Gamma}$.

Consider the edge $e = ((Q_1, a, A_2), b)$ in the price-reward game graph $\widehat{\Gamma}$. The set of inputs $\widehat{\Theta}^{\text{Min}}(e)$, for player Min, is equal to I_3 — the maximum set such that $\mathsf{A}^{\text{MinMax}}(s, I_3) = (a, A_2)$ for every $s \in Q_1$. The set of inputs $\widehat{\Theta}^{\text{Max}}(e)$, for player Max, is $(\mathsf{L} \times V_1) \times [I_3 \rightarrow I_2]$.

To show how the choice of player Max affects the admissible inputs, consider defining inputs for the edge $e' = ((Q_1, a, A_2), a)$, instead of the edge $e = ((Q_1, a, A_2), b)$ — Max chooses control action a instead of b . The set of inputs $\widehat{\Theta}^{\text{Min}}(e')$, for player Min, is equal to I_3 — the same as before. However, the set of inputs $\widehat{\Theta}^{\text{Max}}(e')$, for player Max, is $(\mathsf{L} \times V_1) \times \{\text{id}\}$ — the difference is in the function specification; it reflects that, if Max chooses the control action chosen by Min, he has to choose the same time delay as well.

We conclude this example by showing how a choice of player Min affects the input sets. Consider the edge $e'' = ((Q_3, a, A_1), a)$. The set of inputs $\widehat{\Theta}^{\text{Min}}(e'')$ is equal to I_1 — the maximum set such that $\mathsf{A}^{\text{MinMax}}(s, I_1) = (a, A_1)$ for every $s \in Q_3$. The set of inputs $\widehat{\Theta}^{\text{Max}}(e'')$, of player Max, is equal to $(\mathsf{L} \times V_1) \times \{\text{id}\}$ — the set A_1 does not contain control actions other than the one chosen by Min. \square

5.2.2 Solving hybrid average-price-per-reward games

This section is committed to showing that hybrid average-price-per-reward games are positionally determined, decidable and admit positional ε -optimal strategies. To

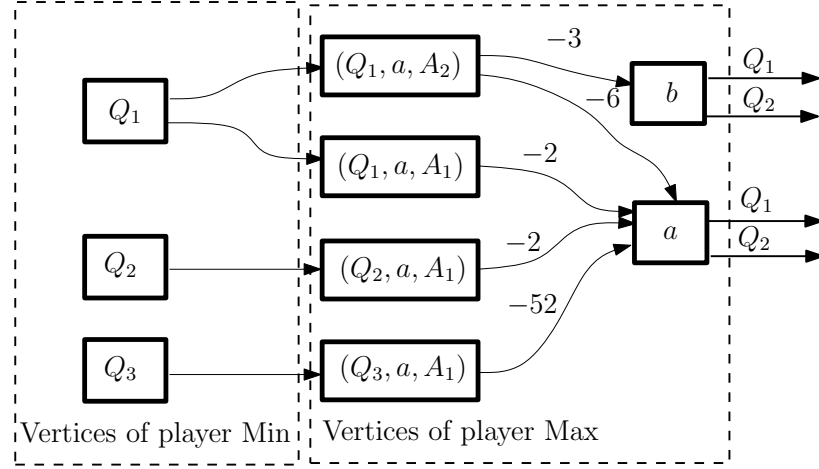


Figure 5.3: The price-reward game graph $\hat{\Gamma}$ obtained, from Γ introduced in Example 5.11, using the finite abstraction \sim . A_1 stands for $\{a\}$ and A_2 for $\{a, b\}$. Edge price is omitted when it is equal to zero. Edges are annotated with constant prices (if omitted, it is equal to 0); the rewards are omitted, as they are constant and equal to 1.

achieve this, we first characterise the game values, of the average-price-per-reward game $\hat{\Gamma}$, using average-price-per-reward optimality equations from Section 2.2.2, and then proceed to prove that solutions to $\text{Opt}_{\text{AvgPR}}(\hat{\Gamma})$ coincide with the solutions to $\text{Opt}_{\text{AvgPR}}(\Gamma)$ (Theorem 5.26). This, together with the results from Section 4.2 proves that hybrid average-price-per-reward games are determined. As a side note, we explain why in the case of average-price games, price-reward graphs with fixed prices and rewards are sufficient.

First, we will present the average-price-per-reward optimality equations in a form that better reflects, compared to the generic form found in Section 4.1.2, the structure of the game graph $\hat{\Gamma}$. For the remainder of this section we assume that \mathcal{R} in $\sim_{\mathcal{R}}$ is equal to $\{R(a) : a \in A\}$, and hence omit it.

For $Q \in S/\sim = \hat{S}^{\text{Min}}$, we have:

$$\begin{aligned} \hat{G}(Q) &= \min \left\{ \hat{G}(Q, a, A) : (Q, (Q, a, A)) \in \hat{E} \right\}, \\ \hat{B}(Q) &= \min \left\{ \hat{B}(Q, a, A) : (Q, (Q, a, A)) \in \hat{E} \text{ and } \hat{G}(Q) = \hat{G}(Q, a, A) \right\}; \end{aligned}$$

for $(Q, a, A) \in (S/\sim \times A \times 2^A) \subseteq \widehat{S}^{\text{Max}}$, we have:

$$\begin{aligned}\widehat{G}(Q, a, A) &= \max \left\{ \widehat{G}(Q, a, A) : ((Q, a, A), a') \in \widehat{E} \right\}, \\ \widehat{B}(Q, a, A) &= \max \left\{ \text{Val} \left(\mathfrak{D}_{((Q, a, A), a')} \left(\widehat{G}(a') \right) \right) + \widehat{B}(a') : \right. \\ &\quad \left. ((Q, a, A), a') \in \widehat{E} \text{ and } \widehat{G}(Q, a, A) = \widehat{G}(a') \right\};\end{aligned}$$

and for $a \in A \subseteq \widehat{S}^{\text{Max}}$, we have:

$$\begin{aligned}\widehat{G}(a) &= \max \left\{ \widehat{G}(Q) : (a, Q) \in \widehat{E} \right\}, \\ \widehat{B}(a) &= \max \left\{ \widehat{B}(Q) : (a, Q) \in \widehat{E} \text{ and } \widehat{G}(a) = \widehat{G}(Q) \right\}.\end{aligned}$$

Recall that only edges of the form $((Q, a, A), a)$ have non-empty input sets. The remaining edges have fixed prices and rewards equal to 0, which gives us:

$$\text{Val} \left(\mathfrak{D}_{(Q, (Q, a, A))} \left(\widehat{G}(Q, a, A) \right) \right) = \text{Val} \left(\mathfrak{D}_{(a, Q)} \left(\widehat{G}(Q) \right) \right) = 0,$$

and hence, we omit these expressions in the equations for $\widehat{B}(Q)$ and $\widehat{B}(a')$.

The game $\mathfrak{D}_{((Q, a, A), a')}(g)$ is in fact a finite duration perfect information game, and therefore it is determined. Determinacy follows from a simple backwards induction argument applied to, the possibly infinite, finite depth game tree.

The main result, Theorem 5.26, states that the solutions to $\text{Opt}_{\text{AvgPR}}(\widehat{\Gamma})$ can be used to obtain the solutions to $\text{Opt}_{\text{AvgPR}}(\Gamma)$.

Theorem 5.26. *Let Γ be a hybrid average-price-per-reward game and let $(\widehat{G}, \widehat{B}) \models \text{Opt}_{\text{AvgPR}}(\widehat{\Gamma})$. If $G, B : S \cup (S \times (A \times T)) \cup A \rightarrow \mathbb{R}$ are such that $G(a) = \widehat{G}(a)$ and $B(a) = \widehat{B}(a)$ for all $a \in A$, and satisfy equations (5.1–5.4), then $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$.*

Corollary 5.27. *Definable average-price-per-reward hybrid games with strong resets are decidable.*

Proof. The price-reward game graph $\hat{\Gamma}$ is finite and definable, hence (\hat{G}, \hat{B}) is definable (Remark 4.8). If Theorem 5.26 holds, by Proposition 5.29, functions $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$ are definable. This, together with Theorem 4.10, yields decidability (Proposition 2.5). \square

Corollary 5.28. *If Γ is a definable hybrid game with strong resets and if in the average-price-per-reward game a player can always make a rational ε -optimal move, then ε -optimal strategies are computable.*

Proof. The proof is similar to the proof of Corollary 4.21. The only difference is that we use Theorem 5.26 (instead of Corollary 4.20) to establish the existence of solutions to the average-price-per-reward optimality equations. Definability of the solutions follows from Proposition 5.29. \square

Our goal is to show that a solution (\hat{G}, \hat{B}) of $\text{Opt}_{\text{AvgPR}}(\hat{\Gamma})$ can be used to obtain a solution (G, B) of $\text{Opt}_{\text{AvgPR}}(\Gamma)$. Recall that a solution of average-price-per-reward optimality equations $\text{Opt}_{\text{AvgPR}}(\Gamma)$ for a hybrid average-price-per-reward game is a pair (G, B) of functions $G, B : \mathbf{S} \cup (\mathbf{S} \times (\mathbf{A} \times \mathbf{T})) \cup \mathbf{A} \rightarrow \mathbb{R}$.

Before we start the proof of Theorem 5.26, we introduce a simple observation, on the structure of the solutions to the average-price-per-reward optimality equations $\text{Opt}_{\text{AvgPR}}(\Gamma)$, which will be helpful in the proof of the actual theorem.

Proposition 5.29. *If Γ is a hybrid average-price-per-reward game, then for all states $s \in \mathbf{S}$ and for all $\tau \in \mathbf{M}^{\text{Min}}(s)$, the values $G(s), B(s), G(s, \tau)$, and $B(s, \tau)$ satisfying equations (5.1–5.4), respectively, are uniquely determined and first-order definable (provided Γ is definable) from the (finitely many) values $\{G(a), B(a) : a \in \mathbf{A}\}$.*

Proof. This is a direct consequence of Remark 4.8. \square

We now proceed to prove Theorem 5.26.

Proof of Theorem 5.26. The proof is based on the non-trivial observation that the \sim equivalence facilitates a separation of discrete and continuous components of each player's moves. Players can first choose their discrete moves in the same order as in a game round, and later choose in a similar fashion corresponding continuous components.

We need to prove that, if for all $a \in \mathbf{A}$, $G(a) = \widehat{G}(a)$ and $B(a) = \widehat{B}(a)$, then indeed $(G, B) \models \text{Opt}_{\text{AvgPR}}(\Gamma)$ (recall that, by Proposition 5.29, this assignment uniquely determines the functions G and B). Average-price-per-reward optimality equations, $\text{Opt}_{\text{AvgPR}}(\Gamma)$, can be expanded so that the conditions for $G(a)$ and $B(a)$ are given in terms of $G|_{\mathbf{A}}$ and $B|_{\mathbf{A}}$, i.e., in terms of the gain and bias functions whose domain is restricted to the set \mathbf{A} . We will show, that the solution (G, B) , postulated in the theorem statement, satisfies the average-price-per-reward optimality equations in the expanded form.

We expand the equation for $\widehat{B}(a)$ with respect to $\text{Opt}_{\text{AvgPR}}(\widehat{\Gamma})$:

$$\begin{aligned}
\widehat{B}(a) &= \max_{\substack{\widehat{G}(a)=\widehat{G}(Q), \\ Q \subseteq \mathbf{R}(a)}} \widehat{B}(Q) \\
&= \max_{\substack{\widehat{G}(a)=\widehat{G}(Q), \\ Q \subseteq \mathbf{R}(a)}} \min_{\substack{\widehat{G}(Q)=\widehat{G}(Q, a', A'), \\ (a', A') \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})}} \widehat{B}(Q, a', A') \\
&= \max_{\substack{\widehat{G}(a)=\widehat{G}(Q), \\ Q \subseteq \mathbf{R}(a)}} \min_{\substack{\widehat{G}(Q)=\widehat{G}(Q, a', A'), \\ (a', A') \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})}} \max_{\substack{\widehat{G}(a'')=\widehat{G}(Q, a', A'), \\ a'' \in A'}} \text{Val} \left(\varrho_{((Q, a', A'), a'')}(\widehat{G}(a'')) \right) + \widehat{B}(a'') \quad (5.11)
\end{aligned}$$

If we expand the definition of $\text{Val}(\varrho_{((Q, a', A'), a'')}(\widehat{G}(a'')))$, we get:

$$\sup_{\substack{(s, f) \in \\ \widehat{\Theta}^{\text{Max}}((Q, a', A'), a'')}} \inf_{\substack{f' \in \\ \widehat{\Theta}^{\text{Min}}((Q, a', A'), a'')}} \pi(s, f(f'(s))) - \rho(s, f(f'(s))) \cdot \widehat{G}(a'').$$

Recall that choosing inputs, in the edge game, is in fact choosing the continuous components of the reconstructed game round (see Section 5.2.1) that match the

discrete ones, encoded in the edge. As a consequence, the expression for the game value can be rewritten as:

$$\sup_{s \in Q} \inf_{\substack{(a', t') \in \mathbf{M}^{\text{Min}}(s), \\ A' = \mathbf{A}^{\text{Max}}(s, (a', t'))}} \sup_{(a'', t'') \in \mathbf{M}^{\text{Max}}(s, (a', t'))} \pi(s, (a'', t'')) - \rho(s, (a'', t'')) \cdot \widehat{G}(a''),$$

which allows us to rewrite expression 5.11, for $\widehat{B}(a)$, as:

$$\begin{aligned} \max_{\substack{Q \subseteq \mathbf{R}(a) \\ \widehat{G}(a) = \widehat{G}(Q)}} \min_{\substack{(a', A') \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T}) \\ \widehat{G}(Q) = \widehat{G}(Q, a', A')}} \max_{\substack{a'' \in A' \\ \widehat{G}(a'') = \widehat{G}(Q, a', A')}} \sup_{s \in Q} \inf_{\substack{(a', t') \in \mathbf{M}^{\text{Min}}(s), \\ A' = \mathbf{A}^{\text{Max}}(s, (a', t'))}} \sup_{(a'', t'') \in \mathbf{M}^{\text{Max}}(s, (a', t'))} \\ \pi(s, (a'', t'')) - \rho(s, (a'', t'')) \cdot \widehat{G}(a'') + \widehat{B}(a''). \end{aligned}$$

Observe that, the choice of $s \in Q$ depends only on Q , and that the choice $(a', t') \in \mathbf{M}^{\text{Min}}(s)$ depends only on $s \in Q$ and $(a', A') \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})$. This allows us to rewrite the last equation in the following way:

$$\begin{aligned} \max_{\substack{Q \subseteq \mathbf{R}(a) \\ \widehat{G}(a) = \widehat{G}(Q)}} \sup_{s \in Q} \min_{\substack{(a', A') \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T}) \\ \widehat{G}(Q) = \widehat{G}(Q, a', A')}} \inf_{\substack{(a', t') \in \mathbf{M}^{\text{Min}}(s), \\ A' = \mathbf{A}^{\text{Max}}(s, (a', t'))}} \sup_{\substack{a'' \in A' \\ \widehat{G}(a'') = \widehat{G}(Q, a', A')}} \sup_{(a'', t'') \in \mathbf{M}^{\text{Max}}(s, (a', t'))} \\ \pi(s, (a'', t'')) - \rho(s, (a'', t'')) \cdot \widehat{G}(a'') + \widehat{B}(a''), \end{aligned}$$

which simplifies to:

$$\sup_{s \in \mathbf{R}(a)} \inf_{\substack{\tau \in \mathbf{M}^{\text{Min}}(s) \\ G(s, \tau) = G(a)}} \max_{\substack{G(a'') = G(s, \tau) \\ (a'', t'') \in \mathbf{M}^{\text{Max}}(s, \tau)}} \pi(s, a'', t'') - \rho(s, a'', t'') \cdot G(s, \tau) + B(a''),$$

as $\widehat{G}(a'') = G(a'')$ and $\widehat{B}(a'') = B(a)$. We can now contract the expression for $\widehat{B}(a)$,

in the following way:

$$\begin{aligned}
\widehat{B}(a) &= \sup_{\substack{G(s)=G(a), \\ s \in R(a)}} \inf_{\substack{G(s,\tau)=G(a), \\ \tau \in M^{\text{Min}}(s)}} B(s, \tau) \\
&= \sup_{\substack{G(s)=G(a), \\ s \in R(a)}} B(s) \\
&= B(a).
\end{aligned}$$

We have shown that the values $\widehat{B}(a)$ and $\widehat{G}(a)$, for all $a \in A$, satisfy the average-price-per-reward optimality equations $\text{Opt}_{\text{AvgPR}}(\Gamma)$, which, together with Proposition 5.29, yields the desired result. \square

Average-price games. As explained earlier (see Section 5.1.1) In an average-price games of a timed action is averaged over the total number of timed actions. In other words, it is an average-price-reward game in which the reward of a timed action is constant and equal to 1. We will now show, that in this special case the input mechanism is not necessary, and that we can reduce to finite doubly-price game graphs.

Let $((Q, a, A), a')$ be an edge bearing a non-zero price (and reward) in the game graph $\widehat{\Gamma}$. Recall that the expression for $\text{Val}(\varnothing_{((Q,a,A),a')}(\widehat{G}(a')))$ is:

$$\sup_{(s,f) \in \widehat{\Theta}^{\text{Max}}((Q,a,A),a')} \inf_{f' \in \widehat{\Theta}^{\text{Min}}((Q,a,A),a')} \pi(s, f(f'(s))) - \rho(s, f(f'(s))) \cdot \widehat{G}(Q, a, A),$$

however, the reward is equal 1, for all the inputs, so the expression can be simplified to the following form:

$$\left(\sup_{(s,f) \in \widehat{\Theta}^{\text{Max}}((Q,a,A),a')} \inf_{f' \in \widehat{\Theta}^{\text{Min}}((Q,a,A),a')} \pi(s, f(f'(s))) \right) - \widehat{G}(Q, a, A).$$

The value of the expression in brackets is well defined, and does not depend on any external parameters. As such, it can be assigned as the the fixed priced of the

edge $((Q, a, A), a')$. This, together with the fact that the reward is fixed, facilitates the reduction from average-price hybrid games to average-price games on doubly-weighted game graphs. The correctness of the reduction follows from Theorem 5.26.

We conclude with an example, which shows how solutions to $\text{Opt}_{\text{AvgPR}}(\hat{\Gamma})$ allow to establish the solutions to $\text{Opt}_{\text{AvgPR}}(\Gamma)$.

Example 5.30. Recall the game graph $\hat{\Gamma}$ from Example 5.25. Fig 5.4 depicts the optimal choices of both players in the average-price game and the solution to the average-price-per-reward optimality equations for finite average-price games. The value of the game from every state is $-2/3$, because $(\hat{G}, \hat{B}) \models \text{Opt}_{\text{AvgPR}}(\hat{\Gamma})$.

We use the solutions to $\text{Opt}_{\text{AvgPR}}(\hat{\Gamma})$ to obtain solutions to $\text{Opt}_{\text{AvgPR}}(\Gamma)$. We set $G \equiv -\frac{2}{3}$, $B(a) = \hat{B}(a) = 0$ and $B(b) = \hat{B}(b) = 0$. The remaining values are uniquely determined by these. One can see that the value of the average-price game on Γ is $-\frac{2}{3}$ and that the players have ε -optimal strategies as follows: from every state in Q_1 player Min should play $(a, 6 - \varepsilon)$, and from every state in $Q_2 \cup Q_3$, Min should play $(a, 2 - \varepsilon)$. Player Max on the other hand has always to play Min's choice unless he is in the state (s, a, t) and $t > 5$, when he should make the move $(b, 3 + \varepsilon)$. From every state in A , Max should choose to play $(0, 0) \in Q_2$. \square

5.2.3 Solving hybrid reachability-price games

In this section we show that hybrid reachability-price games are positionally determined, decidable and admit positional ε -optimal strategies. To achieve this, we first characterise the game values, of the reachability-price game $\hat{\Gamma}$, using reachability-price optimality equations from Section 2.2.2, and then proceed to prove that solutions to $\text{Opt}_{\text{Reach}}(\hat{\Gamma})$ coincide with the solutions to $\text{Opt}_{\text{Reach}}(\Gamma)$ (Theorem 5.31). This, together with the classical results presented in Section 2.2.2 proves that hybrid reachability-price games are determined.

First, we will present the reachability-price optimality equations in a form that better reflects, compared to the generic form found in Section 2.2.2, the struc-

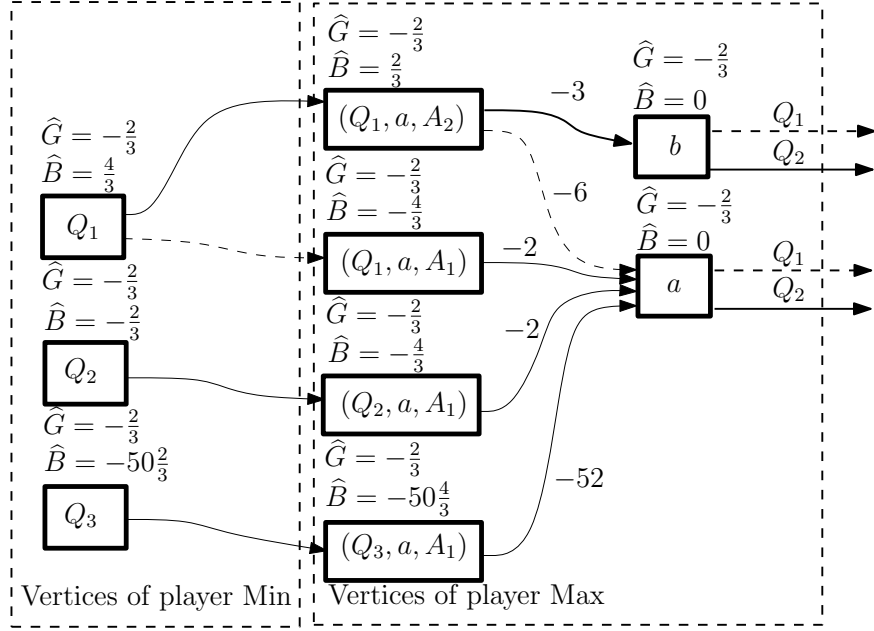


Figure 5.4: Solid arrows denote the optimal strategies of both players. Above each vertex one can find its gain and bias.

ture of the game graph $\hat{\Gamma}$. For the remainder of this section we assume that \mathcal{R} in $\sim_{\mathcal{R}}$ is equal to $\{\mathbf{R}(a) : a \in \mathbf{A}\} \cup \{\mathbf{F}\}$, and hence omit it.

It will be sufficient to consider $\hat{\Gamma}$ with fixed prices and no rewards, i.e., a finite weighted game graph (see Section 2.2.2). Similar approach has been used in the context of average-price games (see the end of Section 5.2.2). The weight function $\hat{\pi} : \hat{\mathbf{E}} \rightarrow \mathbb{R}$ is defined as follows. For $((Q, a, A), a') \in \hat{\mathbf{E}}$ we have:

$$\begin{aligned} \hat{\pi}((Q, a, A), a') &= \sup_{s \in Q} \inf_{t \in \mathbf{T}_{s,(a,A)}^{\text{Min}}} \sup_{t' \in \mathbf{T}_{s,(a,t),a'}^{\text{Max}}} \pi(s, (a', t')), \text{ where} \\ \mathbf{T}_{s,(a,A)}^{\text{Min}} &= \{t \in \mathbf{T} : (a, t) \in \mathbf{M}^{\text{Min}}(s) \text{ and } A = \mathbf{A}^{\text{Max}}(s, (a, t))\}, \\ \mathbf{T}_{s,(a,t),a'}^{\text{Max}} &= \{t' \in \mathbf{T} : (a', t') \in \mathbf{M}^{\text{Max}}(s, (a, t))\}; \end{aligned}$$

and for the remaining edges it is equal to 0.

The following are the reachability-price optimality equations for the finite reachability-price game $(\hat{\Gamma}, \hat{\mathbf{F}})$. If $\hat{s} \in \hat{\mathbf{F}}$ then $\hat{P}(\hat{s}) = \hat{D}(\hat{s}) = 0$. Otherwise, the

following hold. For $Q \in S/\sim = \widehat{S}^{\text{Min}}$, we have:

$$\begin{aligned}\widehat{P}(Q) &= \min \left\{ \widehat{P}(Q, a, A) : (Q, (Q, a, A)) \in \widehat{E} \right\}, \\ \widehat{D}(Q) &= \min \left\{ 1 + \widehat{D}(Q, a, A) : \widehat{P}(Q) = \widehat{P}(Q, a, A) \text{ and } (Q, (Q, a, A)) \in \widehat{E} \right\};\end{aligned}$$

for $(Q, a, A') \in (S/\sim \times A \times 2^A) \subseteq \widehat{S}^{\text{Max}}$, we have:

$$\begin{aligned}\widehat{P}(Q, a, A) &= \min \left\{ \widehat{\pi}((Q, a, A), a') + \widehat{P}(a') : ((Q, a, A), a') \in \widehat{E} \right\}, \\ \widehat{D}(Q, a, A) &= \min \left\{ 1 + \widehat{D}(a') : \right. \\ &\quad \left. \widehat{P}(Q, a, A) = \widehat{\pi}((Q, a, A), a') + \widehat{P}(a') \text{ and } ((Q, a, A), a') \in \widehat{E} \right\};\end{aligned}$$

and for $a \in A \subseteq S^{\text{Max}}$, we have:

$$\begin{aligned}\widehat{P}(a) &= \max \left\{ \widehat{P}(Q) : (a, Q) \in \widehat{E} \right\}, \\ \widehat{D}(a) &= \max_{(a, Q) \in \widehat{E}} \left\{ 1 + \widehat{D}(Q) : \widehat{P}(a) = \widehat{P}(Q) \text{ and } (a, Q) \in \widehat{E} \right\}.\end{aligned}$$

We assume that the value of a reachability-price game in every state of $\widehat{\Gamma}$ is finite (this implies that the values of the hybrid counterpart are finite too). In Section 2.2.2, we have explained how to determine the set of states, \widehat{W}^{Max} , that admit ∞ , and the set of states, \widehat{W}^{Min} , that admit $-\infty$, as game values. By applying Proposition 5.34 (defined later in this section) we can establish definability of the respective sets of states in the hybrid reachability-price game Γ .

As in the case of average-price hybrid games the solutions to the reachability-price optimality equations for the finite game $(\widehat{\Gamma}, \widehat{F})$ coincide with the solutions to the reachability-price optimality equations for the hybrid game (Γ, F) . The main results is as follows, and is proved in a similar fashion as Theorem 5.26.

Theorem 5.31. *For a hybrid reachability-price game Γ , let $(\widehat{P}, \widehat{D}) \models \text{Opt}_{\text{Reach}}(\widehat{\Gamma})$. If for all $a \in A$, it holds that $P(a) = \widehat{P}(a)$ and $D(a) = \widehat{D}(a)$, then $P, D : S \cup (S \times$*

$(A \times T) \cup A \rightarrow \mathbb{R}$ are uniquely determined and $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma)$.

Corollary 5.32. *Definable reachability-price hybrid games with strong resets are decidable.*

Proof. The finite weighted game graph $\hat{\Gamma}$ and the set of goal states \hat{F} are definable, hence (\hat{P}, \hat{D}) is definable (Remark 5.12). If Theorem 5.31 holds, by Proposition 5.34, functions $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma, F)$ are definable. This, together with Theorem 5.14, yields decidability (Proposition 2.5). \square

Corollary 5.33. *If Γ is a definable hybrid game with strong resets and if in the reachability-price game a player can always make a rational ε -optimal move, then ε -optimal strategies are computable.*

Proof. By Corollary 5.15, the existence of $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma)$ implies definability of ε -optimal strategies of both players, provided that F is definable. Theorem 5.31 ensures that such a pair of functions, (P, D) , indeed exists. \square

Our goal is to show that a solution (\hat{P}, \hat{D}) of $\text{Opt}_{\text{Reach}}(\hat{\Gamma}, \hat{F})$ can be used to obtain a solution (P, D) of $\text{Opt}_{\text{Reach}}(\Gamma, F)$. Recall that a solution of reachability-price optimality equations $\text{Opt}_{\text{Reach}}(\Gamma, F)$ for a hybrid reachability-price game is a pair (P, D) of functions $P : S \cup (S \times (A \times T)) \cup A \rightarrow \mathbb{R}$ and $D : S \cup (S \times (A \times T)) \cup A \rightarrow \mathbb{N}$.

Before we start the proof of Theorem 5.31, we introduce a simple observation, on the structure of the solutions to the reachability-price optimality equations $\text{Opt}_{\text{Reach}}(\Gamma, F)$, which will be helpful in the proof of the actual theorem.

Proposition 5.34. *If (Γ, F) is a hybrid reachability-price game then for all states $s \in S$ and for all $\tau \in M^{\text{Min}}(s)$, the values $P(s)$, $D(s)$, $P(s, \tau)$, and $D(s, \tau)$ satisfying equations (5.7–5.10), respectively, are uniquely determined and first-order definable (provided that Γ definable) from the (finitely many) values $\{P(a), D(a) : a \in A\}$.*

Proof. The same as of Proposition 5.29. \square

We now proceed to the proof of Theorem 5.31.

Proof of Theorem 5.31. The proof is based on the non-trivial observation that the \sim equivalence facilities of discrete and continuous components of each player's moves. Players can first choose their discrete moves in the same order as in a game round, and later choose in a similar fashion corresponding continuous components.

We need to prove that, if for all $a \in \mathbf{A}$, $P(a) = \hat{P}(a)$ and $D(a) = \hat{D}(a)$, then indeed $(P, D) \models \text{Opt}_{\text{Reach}}(\Gamma, \mathbf{F})$. (recall that, by Proposition 5.34, this assignment uniquely determines the functions P and D). Reachability-price optimality equations, $\text{Opt}_{\text{Reach}}(\Gamma)$, can be expanded so that the conditions for P and D are given in terms of $P|_{\mathbf{A}}$ and $B|_{\mathbf{A}}$. We will show, that the solution (P, D) , postulated in the theorem statement, satisfies the reachability-price optimality equations in the expanded form.

We start by expanding the equation for $\hat{P}(a)$ with respect to $\text{Opt}_{\text{Reach}}(\hat{\Gamma})$.

$$\begin{aligned}
\hat{P}(a) &= \max_{Q \subseteq \mathbf{R}(a)} \hat{P}(Q) \\
&= \max_{Q \in \mathbf{R}(a)} \min_{\substack{(a', A') \in \\ \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})}} \hat{P}(Q, a', A') \\
&= \max_{Q \in \mathbf{R}(a)} \min_{\substack{(a', A') \in \\ \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})}} \max_{a'' \in A'} \hat{P}(a'') + \hat{\pi}((Q, a', A'), a'') \\
&= \max_{Q \in \mathbf{R}(a)} \min_{\substack{(a', A') \in \\ \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})}} \max_{a'' \in A'} \hat{P}(a'') + \sup_{s \in Q} \inf_{t' \in \mathbf{T}_{s, (a', A')}^{\text{Min}}} \sup_{t'' \in \mathbf{T}_{s, (a', t'), a''}^{\text{Max}}} \pi(s, (a'', t''))
\end{aligned}$$

Notice that, the choice of $s \in Q$ depends only on Q , and that the choice $t \in \mathbf{T}_{s, (a', A')}^{\text{Min}}$ depends only on $s \in Q$ and $(a', A') \in \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})$. This allows us to rewrite the

last equation in the following way (recall that $P(a'') = \widehat{P}(a'')$):

$$\begin{aligned}
\widehat{P}(a) &= \max_{Q \in \mathbf{R}(a)} \sup_{s \in Q} \min_{\substack{(a', A') \in \\ \mathbf{A}^{\text{MinMax}}(Q, \mathbf{T})}} \inf_{t' \in \mathbf{T}_{s, (a', A')}^{\text{Min}}} \max_{a'' \in A'} \sup_{t'' \in \mathbf{T}_{s, (a', t'), a''}^{\text{Max}}} P(a'') + \pi(s, (a'', t'')) \\
&= \sup_{s \in \mathbf{R}(a)} \inf_{\substack{(a', t') \in \\ \mathbf{M}^{\text{Min}}(s, (a', t'))}} \sup_{\substack{(a'', t'') \in \\ \mathbf{M}^{\text{Max}}(s, (a', t'))}} P(a'') + \pi(s, (a'', t'')) \\
&= \sup_{s \in \mathbf{R}(a)} \inf_{\substack{(a', t') \in \\ \mathbf{M}^{\text{Min}}(s, (a', t'))}} P(s, (a', t')) \\
&= \sup_{s \in \mathbf{R}(a)} P(s) \\
&= P(a)
\end{aligned}$$

We deal with the D function in the similar fashion. This allows us to establish that the functions (P, D) , as postulated in the theorem statement, indeed satisfy the reachability-price optimality equations $\text{Opt}_{\text{Reach}}(\Gamma)$. \square

Chapter 6

Conclusions

In this thesis we have considered games for optimal controller synthesis on hybrid automata. In Chapter 3, we have studied reachability-price games on single-clock timed automata. In Chapter 4, we have introduced and studied average-price-per-reward games on a novel class of graphs, the so called price-reward game graphs. In Chapter 5, we have studied average-price-per-reward and reachability-price games on hybrid automata with strong resets.

In Chapter 3, we studied the problem of computing the values for reachability-price games on single-clock timed automata. Our work improves on the earlier results of Bouyer et al. [BLMR06]. For this problem, we have lowered the computational complexity upper-bound from 3EXPTIME to EXPTIME. However, the bound is not known to be tight, as the best lower bound known is PTIME — the complexity of computing the value of a reachability-price games on a finite graph.

There are several possible future research directions. The first, and obvious direction would be to close the complexity gap. We have studied several simple examples which suggest that the problem is EXPTIME-complete, but we did not manage to prove this. The second direction would be to study two-clock timed automata. In this case, the problem of computing the value of a reachability-time game was shown to be EXPTIME-complete [JT07]. However, the problem of computing

the value of a reachability-price game in general remains open. Unfortunately, it is unlikely that the ideas used in Chapter 3 can be used to address this problem, as they rely heavily on the assumption that the automaton has only one clock. The third research direction would be to study reachability-price games on general timed automata. Bouyer et al. [BBM06], prove that the problem of deciding whether there exists an optimal strategy in a reachability-price game on a timed automaton with at least three clocks is undecidable. However, the question whether it is possible to decide the value of the reachability-price game is still open.

The motivation for the work presented in Chapter 4 was the problem of solving average-price-per-reward games on hybrid automata. However, we believe that the class of price-reward game graphs is interesting in its own right, and to the best of our knowledge, this model has not been considered before. To establish our results that average-price-per-reward games on price-reward game graphs are determined, we use a combination of two techniques: optimality equations and strategy improvement. These techniques have been known in the literature [JT07; BV07; Put94; FV97] but they were used differently — optimality equations were used to prove that a strategy improvement algorithm indeed computes the value of a game. We have used the fact that the values of an average-price-per-reward game, if they exist, are characterised using optimality equations, to prove determinacy (i.e., existence of game values). Strategy improvement enabled us to lift the determinacy of average-price-per-reward games on graphs of out-degree one, to all graphs. We believe this approach is novel, and could be applied to other problems.

In Chapter 5, we have proved that average-price-per-reward and reachability-price games on hybrid systems with strong resets are determined. To achieve this, we have reduced the problem of solving a hybrid game to a problem of solving a finite game. In the case of average-price-per-reward games, we reduced to the analogous problem on finite price-reward graphs (introduced in Chapter 4), and in the case of reachability-price games we reduced to the analogous problem on finite weighted

graphs.

When hybrid automata with strong resets were first introduced [LPS00; BM05], they were advertised as allowing for rich continuous dynamics at the cost of the strong reset property. The strong resets made decoupling of discrete and continuous components possible. Additionally, the fact that continuous components were defined using first-order logic over an o-minimal structure, was used to prove existence of finite bisimulations [LPS00; BM05]. The o-minimality requirement was crucial in establishing these results. In this context the semantics of the automaton allowed for an arbitrary number of continuous transitions prior to a discrete one. This is in contrast to the game setting, where like in this thesis, an execution of the hybrid automaton is an alternating sequence of continuous and discrete transitions. In the game setting, the emphasis is placed on the interaction of the controller and the environment, rather than on the rich continuous dynamics of the continuous components [BBJ⁺08; BBC06; BBC09]. This is reflected by the timed-action semantics and the steps of the game round. This change of emphasis reduced the continuous behaviours of the automaton to “parameters” of an execution that determine its quantitative properties. For average-price and reachability-price games on hybrid systems with strong resets, the optimal “parameters” can be determined a priori — there is a reduction from hybrid games to their finite counterparts [BBJ⁺08]. Average-price-per-reward games are more complex, and the optimal choice of “parameters” depends on the discrete behaviour chosen by the players. This is the reason for reducing to average-price-per-reward games on finite price-reward graphs.

In the light of the above, there are two possible directions for future research. One is to look for new ways of modelling the controller-environment interactions that allow for richer continuous behaviours. The other direction is to work on models without the strong reset property, e.g., STORMED hybrid automata [VPVD08; VPVD09].

Appendix A

Proof of Theorem 3.10

The proof of the theorem follows from the Lemmas A.1–A.3. The first lemma establishes an exponential time Turing-reduction from single-clock timed automata to single-clock timed automata without positive prices on control actions. The second lemma establishes a further polynomial-time Turing-reduction to $[0, 1]$ -bounded single-clock timed automata. The final lemma establishes a further polynomial time Turing-reduction to simple timed automata.

We now proceed to prove the aforementioned Lemmas A.1–A.3, which establish that restricting to simple timed automata can be done without loss of generality.

Lemma A.1. *The problem of deciding the value of a reachability-price game on single-clock timed automata is exponential time Turing-reducible to the analogous problem on single-clock timed automata without positive prices on control actions.*

Proof. We assume that reachability-price games on single-clock timed automata without positive prices on control actions are determined, that the Val function for such games is computable, and that it is a quasi-cost function at every control location.

To prove the lemma, we will construct a sequence of reachability-price games based on the original game Γ , but defined on single-clock timed automata without

positive prices on control actions. The game value of the i -th game in the sequence will denote the value of the original game Γ , provided that the players were allowed to take at most i discrete transitions that correspond to control actions with a positive price. The $(i+1)$ -th game will be defined in terms of the original game and the Val function of the i -th game; its size will be polynomially larger than the size of the original game Γ . To decide the value of a reachability-price game, it will be sufficient to consider only a finite number of elements in this sequence since every control action with a positive price has a price of at least one.

We start by defining the first element of the aforementioned sequence. The initial game will be played on the automaton with all control actions admitting positive prices removed — by assumptions, we can compute the Val function for such games.

Given a reachability-price game Γ defined on an automaton \mathcal{A} , we define a new game Γ_0 that is played on an automaton \mathcal{A}_0 , obtained from the original one by removing all control actions that admit a positive price, i.e., price of 1 or greater. All components of the automaton \mathcal{A}_0 are inherited from the automaton \mathcal{A} , with the exception of the set of control actions, which is defined as follows.

$$A^0 = \left\{ a : a \in A \text{ and } \pi(a) = 0 \right\}.$$

The set of control locations in the automaton \mathcal{A}^0 is inherited from the automaton \mathcal{A} , and as a consequence the definition of the game Γ^0 is the same as the definition of the original game (recall that the definition of a game consists of: a partition of the set of control locations, a set of goal locations, and a function that assigns a quasi-cost function to every goal location).

We now explain how to construct the $(i+1)$ -th game, given that we have computed the Val function for the i -th game in the sequence. Once again, both the i -th and $(i+1)$ -th game are without discrete transitions admitting positive

prices, and hence, by assumptions, we can compute the Val function for such games. Intuitively, we take the automaton \mathcal{A} and remove all control actions that have positive prices. For every removed control action, which led from control location ℓ to control location ℓ' , we add a new goal location, and a new control action that leads from control location ℓ to this new goal location. The quasi-cost function in this new goal location encodes the quasi-cost of reaching goal from ℓ' in Γ^i , increased by the price of the control action.

Let Γ^i denote the reachability-price game defined over the automaton \mathcal{A}^i . We define the automaton \mathcal{A}^{i+1} and the game Γ^{i+1} , assuming that we have computed Val_{Γ^i} . As starting points, we take the games Γ and Γ^0 , and automata \mathcal{A} and \mathcal{A}^0 . The timed automaton $\mathcal{A}^{i+1} = \langle \{x\}, \mathbf{L}^{i+1}, \mathbf{A}^{i+1}, \text{inv}^{i+1}, \text{urg}^{i+1}, \pi^{i+1} \rangle$ is defined in terms of the previously introduced automata \mathcal{A} and \mathcal{A}^0 . The set of control locations \mathbf{L}^{i+1} is defined as:

$$\mathbf{L}^{i+1} = \mathbf{L} \cup \left\{ a : a \in \mathbf{A} \text{ and } \pi(a) > 0 \right\};$$

the set of control actions, \mathbf{A}^{i+1} , is defined as:

$$\mathbf{A}^{i+1} = \mathbf{A}^0 \cup \left\{ (\ell, I, Z, a) : \ell \in \mathbf{L} \text{ and } a = (\ell, I, Z, \ell') \in \mathbf{L}^{i+1} \cap \mathbf{A} \right\};$$

the invariant condition, $\text{inv}^{i+1} : \mathbf{L}^{i+1} \rightarrow \mathcal{B}([0, M])$, is defined as:

$$\text{inv}^{i+1}(\ell) = \begin{cases} \text{inv}(\ell'') & \text{if } \ell = (\ell', I, Z, \ell'') \in \mathbf{L}^{i+1} \cap \mathbf{A} \\ \text{inv}(\ell) & \text{if } \ell \in \mathbf{L}; \end{cases}$$

the urgency mapping, $\text{urg}^{i+1} : \mathbf{L}^{i+1} \rightarrow \{0, 1\}$, is defined as:

$$\text{urg}^{i+1}(\ell) = \begin{cases} 0 & \text{if } \ell \in \mathbf{L}^{i+1} \cap \mathbf{A} \\ \text{urg}(\ell) & \text{if } \ell \in \mathbf{L}; \end{cases}$$

and the price function, $\pi^{i+1} : \mathsf{L}^{i+1} \rightarrow \mathbb{N}$, is defined as:

$$\pi^{i+1}(\ell) = \begin{cases} 0 & \text{if } \ell \in \mathsf{L}^{i+1} \cap \mathsf{A} \\ \pi(\ell) & \text{if } \ell \in \mathsf{L}. \end{cases}$$

The reachability-price game, $\langle \Gamma^{i+1}, \mathsf{F}^{i+1}, \pi^{\mathsf{F}^{i+1}} \rangle$, on the single-clock timed automaton \mathcal{A}^{i+1} , is defined as in Definition 3.6, according to the following specification: the set of control locations is partitioned as follows $\mathsf{L}^{\text{Min}^{i+1}} = \mathsf{L}^{\text{Min}}$ and $\mathsf{L}^{\text{Max}} = \mathsf{L}^{i+1} \setminus \mathsf{L}^{\text{Min}^{i+1}}$; the set of goal locations is defined as $\mathsf{L}^{\mathsf{F}^{i+1}} = \mathsf{L}^{\mathsf{F}} \cup (\mathsf{L}^{i+1} \cap \mathsf{A})$; and the function, $\text{CF}^{\mathsf{F}^{i+1}} : \mathsf{L}^{\mathsf{F}^{i+1}} \rightarrow \mathcal{CF}([0, M])$, that assigns a quasi-cost function to every goal location is defined as:

$$\text{CF}^{\mathsf{F}^{i+1}}(\ell) = \begin{cases} x \mapsto \text{Val}_{\Gamma^i}((\ell'', 0)) + \pi(\ell) & \text{if } \ell = (\ell', I, \{x\}, \ell'') \in \mathsf{L}^{i+1} \cap \mathsf{A} \\ x \mapsto \text{Val}_{\Gamma^i}((\ell'', x)) + \pi(\ell) & \text{if } \ell = (\ell', I, \emptyset, \ell'') \in \mathsf{L}^{i+1} \cap \mathsf{A} \\ \pi^{\mathsf{F}}(\ell) & \text{if } \ell \in \mathsf{L}^{\mathsf{F}}. \end{cases}$$

Let c be the real constant mentioned in the definition of the reachability-price decision problem. In order to decide, whether $\text{Val}_{\Gamma}(s) \leq c$ for every state s , it suffices to compute the function $\text{Val}_{\Gamma_{c^*}}$, where $c^* = \min\{c' : c' \geq c \text{ and } c' \in \mathbb{N}\}$ — recall that if the price of a control action is positive it is at least 1. This establishes the EXPTIME complexity of the reduction (constants are encoded in binary). \square

Lemma A.2. *The problem of deciding the game value of a reachability-price game on single-clock timed automata, without positive prices on control actions is polynomial time Turing-reducible to the analogous problem on $[0, 1]$ -bounded single-clock timed automata without positive prices on control actions.*

Proof. Let \mathcal{A} be an arbitrary bounded timed automaton without positive prices on control actions — in this context it is natural to assume that the price function has the type $\mathsf{L} \rightarrow \mathbb{N}$, as opposed to $\mathsf{L} \cup \mathsf{A} \rightarrow \mathbb{N}$ in the general definition. In this

proof we assume that reachability-price games on $[0, 1]$ -bounded single-clock timed automata are determined. To prove the lemma, we will construct a sequence of timed automata, which are mutually equivalent, with the final automaton being $[0, 1]$ -bounded. With this hindsight in mind, it will be justified to assume that reachability-price games on single-clock timed automata without positive prices on control actions are determined.

For the purpose of this proof, we will use a notion of equivalence that relates to the game value. More formally, two timed automata \mathcal{A} and \mathcal{A}' are equivalent if there exist two functions $h_1 : S \rightarrow S'$ and $h_2 : S' \rightarrow S$ and two reachability-price games Γ and Γ' , such that $\text{Val}_\Gamma(s) = \text{Val}_{\Gamma'}(h_1(s))$ and $\text{Val}_{\Gamma'}(s') = \text{Val}_\Gamma(h_2(s'))$ for every $s \in S$ and $s' \in S'$, where Γ and Γ' are reachability-price games on \mathcal{A} and \mathcal{A}' respectively. To show such equivalence, it is sufficient to show that strategies from one game can be adapted to the other one, and that they yield the same results.

Before we start the actual proof, we will introduce two auxiliary notions, which will simplify the presentation. Let $I = [b, e]$ be an interval. Unless we clearly state otherwise, it can be open on either of its endpoints; we will write \bar{I} to denote the closure of the interval I , i.e., the smallest closed interval containing I . If we construct a new interval I' based on I , unless stated otherwise, it inherits the “openness” properties of I . We also introduce a special function $\delta : \mathbb{R}_{\geq 0} \rightarrow \mathbb{R}_{\geq 0}$ defined, as follows:

$$\delta(x) = \begin{cases} x & \text{if } x < 1, \\ 1 & \text{otherwise.} \end{cases}$$

Finally, consider the set, $C = \{c_0, c_1, \dots, c_n\}$, of all the constants appearing in the guards of control actions and in the invariant condition, i.e., the set of interval endpoints. Without loss of generality, we assume that $c_0 = 0$, and that $c_i < c_{i+1}$ for every $i \in \{0, \dots, n-1\}$.

The purpose of the first construction is to obtain an automaton that has the

following property: for every two control actions $(\ell_1, I_1, Z_1, \ell'_1)$ and $(\ell_2, I_2, Z_2, \ell'_2)$ we have either $\overline{I_1} = \overline{I_2}$, or $\overline{I_1} \cap \overline{I_2} = \emptyset$, or $\overline{I_1} \cap \overline{I_2} = \{c\}$, for some $c \in C$. For future reference, we will call such a single-clock timed automaton a *type-1 timed automaton*.

We obtain the type-1 timed automaton \mathcal{A}' by redefining the set of control actions in \mathcal{A} . For every control action $(\ell, [b, e], Z, \ell') \in \mathbf{A}$, consider $\{b = c_{i_0}, \dots, c_{i_k} = e\} = C \cap \overline{[b, e]}$, and assume that $c_{i_j} \leq c_{i_{j+1}}$ for every $j \in \{0, \dots, k-1\}$. If $k > 1$, we define \mathbf{A}' to include every (ℓ, I_j, Z, ℓ') , where $I_j = [c_{i_j}, c_{i_{j+1}}]$, for every $j \in \{0, \dots, k-1\}$. Moreover, the “left openness” of the interval I_0 , and “right openness” of I_k are consistent with that of $[b, e]$; on the remaining interval end-points, the I_j intervals are closed. Otherwise, if $k = 1$, \mathbf{A}' contains just the original transition of \mathcal{A} .

The automaton \mathcal{A}' is a type-1 timed automaton, and it is polynomially larger than \mathcal{A} ; the transitions systems of both automata have the same state space. Moreover, from every state the set of reachable states and the price of reaching them are the same as in \mathcal{A} . The definition of the game relies solely on the partition of the set of control locations, so we define Γ' in the same way as Γ . We use the identity mapping to establish equivalence of \mathcal{A} and \mathcal{A}' .

The purpose of the next construction is to obtain a type-1 timed automaton equivalent to the original, that has a transition system with a special property. The property states that there exists a function $f : \mathbf{L} \rightarrow \{I_0, \dots, I_{n-1}\}$ such that, if (ℓ, x) is the state of the transition system, then $x \in f(\ell)$. In other words, for every state the control location encodes the closed interval $I_i = [c_i, c_{i+1}]$, for $i \in 0, \dots, n-1$, to which the clock value belongs. A single-clock timed automaton with this property will be referred to as a *type-2 timed automaton*.

We construct the type 2-timed automaton \mathcal{A}' from a type-1 timed automa-

ton \mathcal{A} in the following way. The set of control locations is defined as follows

$$\mathbf{L}' = \bigcup_{\ell \in \mathbf{L}} \{(\ell, I_i) : 0 \leq i < n - 1\},$$

and the set of control actions is defined as follows

$$\begin{aligned} \mathbf{A}' = & \left\{ ((\ell, I_i), I, \emptyset, (\ell', I_i)) : (\ell, I, \emptyset, \ell') \in \mathbf{A} \text{ and } I \subseteq I_i \right\} \cup \\ & \left\{ ((\ell, I_i), I, \{x\}, (\ell', I_0)) : (\ell, I, \{x\}, \ell') \in \mathbf{A} \text{ and } I \subseteq I_i \right\} \cup \\ & \left\{ ((\ell, I_i), I_i \cap I_{i+1}, \emptyset, (\ell, I_{i+1})) : 0 \leq i \leq n - 2 \right\}. \end{aligned}$$

We define the invariant condition $\text{inv}((\ell, I))$ as $\text{inv}(\ell) \cap I$, the urgency mapping $\text{urg}'((\ell, I))$ as $\text{urg}(\ell)$, and the price function $\pi'((\ell, I))$ as $\pi(\ell)$.

The timed automaton \mathcal{A}' is clearly type-1, and, due to the definition of the invariant condition, it indeed does encode the interval I_i within the control location — the requested function can be trivially defined as $f((\ell, I)) = I$. Additionally, the timed automaton \mathcal{A}' is only polynomially larger than the original timed automaton \mathcal{A} .

To define the game Γ' , we partition \mathbf{L}' into the sets of control locations of player Min and player Max, according to the partition of \mathbf{L} , i.e., the membership of (ℓ, I) in one of the two sets depends on the membership of ℓ in those sets in the game Γ . The set of goal locations is determined in a similar manner. The assignment of quasi-cost functions to goal locations, $\text{CF}^{F'}((\ell, I))$, is defined as $\text{CF}^F(\ell)|_I$, for every $(\ell, I) \in \mathbf{L}'$.

Consider the mapping $h_2 : \mathbf{S}' \rightarrow \mathbf{S}$, defined as follows: $h_2((\ell, I), x) = (\ell, x)$. Given a state $s \in \mathbf{S}'$, for every transition $s \xrightarrow{\theta} s'$ in \mathcal{A}' there exists a transition $h_2(s) \xrightarrow{\theta'} h_2(s')$ in \mathcal{A} that has the same price. Conversely, consider a mapping $h_1 : \mathbf{S} \rightarrow \mathbf{S}'$ that satisfies the following: $h_1((\ell, x)) = ((\ell, I), x)$, where $x \in I$. For every $s \in \mathbf{S}$ and every transition $s \xrightarrow{\theta} s'$, with a price p , there exists a sequence of

transitions $h_1(s) \xrightarrow{\theta_1} s_1 \dots s_{k-1} \xrightarrow{\theta_k} h_1(s')$ in \mathcal{A}' with the same total accumulated price p , where $s_j = ((\ell, I_{i_j}), x_i)$, for every $1 \leq j < k$ (note that all s_j belong the set of states of the same player). We then trivially have that $\text{Val}_{\Gamma'}(((\ell, I), x)) = \text{Val}_{\Gamma}(h_2(((\ell, I), x)))$ and $\text{Val}_{\Gamma}((\ell, x)) = \text{Val}_{\Gamma'}(h_1((\ell, x)))$. This implies equivalence of \mathcal{A} and \mathcal{A}' .

We can further convert a type-2 timed automaton \mathcal{A} , to obtain an automaton \mathcal{A}' that has the following property: in a control location $(\ell, [b, e])$ the clock value always satisfies the clock constraint $[0, e - b]$. Such a single-clock timed automaton will be referred to as *type-3 timed automaton*. In a type-3 time automaton, the control location encodes the interval, and the valuation of the clock encodes the value of the clock relative to the interval's beginning, i.e., a state $((\ell, [b, e]), x)$ of $\mathcal{T}_{\mathcal{A}}$ corresponds to a state $((\ell, [b, e]), x - b)$ in $\mathcal{T}_{\mathcal{A}'}$.

We construct \mathcal{A}' by redefining the set of control actions and the invariant condition of \mathcal{A} . Given an interval $I = [b, e]$, we write $I - c$ to denote the interval $[b - c, e - c]$. For every control location $(\ell, [b, e])$ we define $\text{inv}'((\ell, [b, e])) = \text{inv}((\ell, [b, e])) - b$. Furthermore, the new set of control actions is defined as follows:

$$\begin{aligned} \mathbf{A}' = & \left\{ ((\ell, [b, e]), I'' - b, \{x\}, (\ell', I')) : ((\ell, [b, e]), I'', Z, (\ell', I')) \in \mathbf{A} \text{ and } I \neq I' \right\} \\ & \cup \left\{ ((\ell, [b, e]), I'' - b, \emptyset, (\ell', [b, e])) : ((\ell, [b, e]), I'', \emptyset, (\ell', [b, e])) \in \mathbf{A} \right\}, \end{aligned}$$

where $Z \subseteq \{x\}$. The set of control locations in the timed automaton \mathcal{A}' is the same as in the timed automaton \mathcal{A} , and hence Γ' inherits the partition of control locations and the set of goal locations form the game Γ . To complete the definition of Γ' we define the mapping from goal locations to quasi-cost functions $\text{CF}^{\mathbf{F}'}$ as:

$$(\ell, [b, e]) \mapsto \left(x \mapsto \text{CF}^{\mathbf{F}}((\ell, [b, e]))(x + b) \right).$$

We then have:

$$\text{Val}_\Gamma(((\ell, [b, e]), x)) = \text{Val}_{\Gamma'}(((\ell, [b, e]), x - b)).$$

This establishes the equivalence of \mathcal{A} and \mathcal{A}' , as the mapping $x \mapsto x - b$ is a bijection. Furthermore, the size of \mathcal{A}' is polynomially larger than the size of \mathcal{A} .

In a type-3 timed automaton every control action that leads to a control location that encodes a different interval contains a reset. Moreover, every control action originating from a control location $(\ell, [b, e])$ has a guard equal to $[0, 0]$, $[0, e]$, or $[e, e]$. In the next transformation, we will transform a type-3 timed automaton \mathcal{A} into an equivalent $[0, 1]$ -bounded timed automaton \mathcal{A}' . Every interval $[0, e]$ will be encoded in the interval $[0, 1]$. To maintain equivalence, the price of a control location will be multiplied by the length of the original interval, encoded by that control location.

We will use the δ function, defined in the beginning of the proof, to aid the definition of \mathcal{A}' . The set of control locations, and the urgency mapping remain unchanged. We define the new set of control actions as follows:

$$\mathbf{A}' = \left\{ (\ell, [\delta(b), \delta(e)], Z, \ell') : (\ell, [b, e], Z, \ell') \in \mathbf{A}' \right\}.$$

For every control location $(\ell, I) \in \mathbf{L} = \mathbf{L}'$, we set $\text{inv}'((\ell, I)) = [0, \delta(e)]$, where $\text{inv}((\ell, I)) = [0, e]$, and we define the price function $\pi'((\ell, [b, e]))$ as $(e - b) \cdot \pi((\ell, [b, e]))$.

As in the transformation from a type 2 to a type 3 timed automaton, we have not altered the set of control locations, so the game Γ' inherits the partition of control locations from Γ . We only need to redefine the mapping from goal locations to quasi-cost functions in the following way:

$$\text{CF}^{\mathbf{F}'}((\ell, [b, e]))(x) = \text{CF}^{\mathbf{F}}((\ell, [b, e]))((e - b) \cdot x).$$

The $[0, 1]$ -bounded timed automaton \mathcal{A}' is only polynomially larger than \mathcal{A} .

Moreover we have:

$$\text{Val}_\Gamma(((\ell, [b, e]), x)) = \text{Val}_{\Gamma'}\left(\left((\ell, [b, e]), \frac{x}{e-b}\right)\right),$$

which establishes the equivalence of \mathcal{A} and \mathcal{A}' .

To summarise, given an arbitrary bounded timed automaton \mathcal{A} without positive prices on control actions, using the four transformations we have constructed a polynomially larger $[0, 1]$ -bounded timed automaton \mathcal{A}' without positive prices on control actions, which is equivalent to \mathcal{A} . This finishes the proof. \square

Lemma A.3. *The problem of deciding the value of a reachability-price game on single-clock $[0, 1]$ -bounded timed automata, without positive prices on control actions, is polynomial time Turing-reducible to the analogous problem on simple single-clock timed automata.*

Proof. We assume that reachability-price games on simple-single-clock timed automata are determined. The proof that this is indeed the case will be provided in Section 3.3.

In a transition system of a single-clock timed automaton, the set of states reachable through a clock resetting transition, i.e., a transition labelled with a following label $(\ell, I, \{x\}, \ell')$, is finite. If \mathbf{L} is the set of control locations, then this set (of states) is contained in $\mathbf{L} \times \{0\}$. This implies that if an automaton has n clock resetting transitions, then every run consisting of at least $n + 1$ such transitions has to revisit some state at least once. Furthermore, since all the prices are non-negative, visiting a non-goal state more than once is not beneficial for player Min in a reachability-price game. If, however, player Max can enforce such a multiple visit, then he can enforce a cycle that does not visit a goal state. The proof that restricting to simple timed automata can be done without loss of generality will exploit this intuition.

Given a $[0, 1]$ -bounded timed automaton \mathcal{A} with n clock-resetting transitions,

we will construct a special timed automaton \mathcal{A}' that will use control locations to count the number of resetting transitions in a run of its transition system. Once the count exceeds n , the automaton will enter a special state. In every other respect, this new automaton will behave exactly like the original one. The game Γ' on, \mathcal{A}' , will be naturally defined in terms of the game Γ , on \mathcal{A} , as we will explain further in this sketch of a proof. The only difference is the designated r control location, to which $\text{CF}^{\text{F}'}$ assigns a quasi-cost function equivalent to ∞ .

The automaton \mathcal{A}' will consist of $n + 1$ “copies” of \mathcal{A} . Intuitively, the game starts in the first copy, and each time a clock resetting transition occurs, it progresses to the next copy. If the game is in the $(n + 1)$ -th copy, and a resetting transition occurs, then it enters the special goal state mentioned earlier. There are no clock resetting transitions within a single copy — a simple timed automaton.

It will be clearly visible that the value of the game from the states belonging to the first “copy” of \mathcal{A} , in the game Γ' , is equal to the value of the game Γ . In order to compute the value of the game for the i -th copy (for $i \in \{1, \dots, n\}$), we first compute it for the $(i + 1)$ -th copy, and then alter the i -th copy so that the clock resetting transitions lead to specially created goal locations. The quasi-cost functions, assigned to those control locations, encode the value of the game from relevant control locations in the $(i + 1)$ -th copy.

Formally, we define \mathcal{A}' in the following way:

$$\mathbf{L}' = \mathbf{L} \times \{0, \dots, n\} \cup \{r\}, \text{ where } r \notin \mathbf{L}.$$

The distinguished control location r is the aforementioned special control location that is reached after the $(n + 1)$ -th clock resetting transition. The set of control

actions is defined as:

$$\begin{aligned} A' = & \left\{ ((\ell, i), I, \emptyset, (\ell', i)) : (\ell, I, \emptyset, \ell') \in A \text{ and } 0 \leq i \leq n \right\} \\ & \cup \left\{ ((\ell, i), I, \{x\}, (\ell', i+1)) : (\ell, I, \{x\}, \ell') \in A \text{ and } 0 \leq i < n \right\} \\ & \cup \left\{ ((\ell, n), I, \{x\}, r) : (\ell, I, \{x\}, \ell') \in A \right\} \cup \{(r, [0, 1], \{x\}, r)\}. \end{aligned}$$

For every control location (ℓ, i) , we define the invariant condition as $\text{inv}'((\ell, i)) = \text{inv}(\ell)$, the urgency mapping as $\text{urg}((\ell, i)) = \text{urg}(\ell)$, and the price function as $\pi'((\ell, i)) = \pi(\ell)$.

We now give the definition of the game Γ' in terms of the game Γ . Given a control location $(\ell, i) \in L'$ its membership in $L^{F'}$, $L^{\text{Min}'}$ and $L^{\text{Max}'}$ depends on the membership of ℓ in L^F , L^{Min} and L^{Max} . For every goal location (ℓ, i) , we define $\text{CF}^{F'}((\ell, i)) = \text{CF}^F(\ell)$. Finally, we set $\text{inv}'(r) = [0, 1]$, and $\text{CF}^{F'}(r) \equiv \infty$.

We now show how to compute $\text{Val}_{\Gamma'}$ in the i -th component, provided that we have computed it in the $(i+1)$ -th component. Let $\{a_1, \dots, a_k\} \subseteq A'$, where $a_j = ((\ell_j, i), I_j, \{x\}, (\ell'_j, i+1))$, for $j \in \{1, \dots, k\}$, be the set of all control actions leading from the i -th to the $(i+1)$ -th copy of \mathcal{A} , in A' . We will substitute every clock resetting control action with a control action leading to special goal location, whose quasi-cost function encodes the value of the game should the actual clock resetting control action had been taken.

Let Γ'' be the natural restriction of Γ' to the automaton \mathcal{A}'' whose set of control locations is restricted to $L' \cap (L \times \{i+1\})$, and let

$$g_j(x) = \text{Val}_{\Gamma''}(((\ell'_j, i+1), 0))$$

We define the simple timed automaton \mathcal{A}''' and the reachability-price game Γ''' in

the following way:

$$\mathbf{L}''' = \mathbf{L}'' \cup \{(\ell'_1, i+1), \dots, (\ell'_k, i+1)\},$$

with the partition of \mathbf{L}''' , into the sets of control locations of players Min and Max, being consistent with that of \mathbf{L}'' . Additionally, we define $\mathbf{L}^{F''}$ as $\mathbf{L}^{F''} \cap \mathbf{L}''' \cup \{\ell'_1, \dots, \ell'_k\}$.

We define the set of control actions as

$$\mathbf{A}''' = \mathbf{A}'' \cup \{((\ell_j, i), I, \emptyset, (\ell'_1, i+1)) : 1 \leq j \leq k\}.$$

With inv''' , urg''' , π''' , and $\text{CF}^{F''}$ being consistent with their counterparts in \mathcal{A}'' and Γ'' over \mathbf{L}'' . Additionally, $\text{CF}^{F''}((\ell'_j, i+1)) = g_j$, and $\text{inv}'''((\ell'_j, i+1)) = \text{inv}'((\ell'_j, i+1))$, for $j \in \{1, \dots, k\}$. \square

Bibliography

- [ABP04] R. Alur, M. Bernadsky, and Madhusudan P. Optimal reachability for weighted timed games. In *ICALP'04*, volume 3142 of *LNCS*, pages 122–133. Springer, 2004.
- [ACH⁺95] R. Alur, C. Courcoubetis, N. Halbwachs, T. A. Henzinger, Ho P.-H., X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.
- [ACH97] R. Alur, C. Courcoubetis, and T. A. Henzinger. Computing accumulated delays in real-time systems. *Formal Methods in System Design*, 11:137–155, 1997.
- [ACHH93] R. Alur, C. Courcoubetis, T. A. Henzinger, and P. Ho. Hybrid automata: An algorithmic approach to the specification and verification of hybrid systems. In *Hybrid Systems*, volume 736 of *LNCS*, pages 209–229. Springer, 1993.
- [AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
- [AdAF05] B. T. Adler, L. de Alfaro, and M. Faella. Average reward timed games. In *FORMATS'05*, volume 3829 of *LNCS*, pages 65–80. Springer, 2005.

- [ADG97] M. Antoniotiz, A. Deshpande, and A. Girault. Microsimulation analysis of a hybrid system model of multiple merge junction highway and semi-automatic vehicles. In *ITSC'97*, pages 147–152. IEEE, 1997.
- [AG11] K. R. Apt and E. Grädel, editors. *Lectures in Game Theory for Computer Scientists*. Cambridge University Press, 2011.
- [AHH96] R. Alur, T. A. Henzinger, and P.-H. Ho. Automatic symbolic verification of embedded systems. *IEEE Transactions on Software Engineering*, 22:181–201, 1996.
- [ALP01] R. Alur, S. La Torre, and G. J. Pappas. Optimal paths in weighted timed automata. In *HSCC'01*, volume 2034 of *LNCS*, pages 49–62. Springer, 2001.
- [AM99] E. Asarin and O. Maler. As soon as possible: Time optimal control for timed automata. In *HSCC'99*, volume 1569 of *LNCS*, pages 19–30. Springer, 1999.
- [AMP95] E. Asarin, O. Maler, and A. Pnueli. Symbolic controller synthesis for discrete and timed systems. In *Hybrid Systems II*, volume 999 of *LNCS*, pages 1–20. Springer, 1995.
- [ATP04] R. Alur, S. Torre, and G. J. Pappas. Optimal paths in weighted timed automata. *Theoretical Computer Science*, 318:297–322, 2004.
- [BBBR07] P. Bouyer, T. Brihaye, V. Bruyère, and J.-F. Raskin. On the optimal reachability problem on weighted timed automata. *Formal Methods in System Design*, 31(2):135–175, October 2007.
- [BBC06] P. Bouyer, T. Brihaye, and Fabrice Chevalier. Control in o-minimal hybrid systems. In *LICS'06*, pages 367–378. IEEE Computer Society Press, 2006.

- [BBC07] P. Bouyer, T. Brihaye, and F. Chevalier. Weighted o-minimal hybrid systems are more decidable than weighted timed automata! In *LICS'07*, volume 4514 of *LNCS*, pages 69–83. Springer, 2007.
- [BBC09] P. Bouyer, T. Brihaye, and F. Chevalier. Weighted o-minimal hybrid systems. *Annals of Pure and Applied Logics*, 161(3):268–288, December 2009.
- [BBC10] P. Bouyer, T. Brihaye, and F. Chevalier. O-minimal hybrid reachability games. *Logical Methods in Computer Science*, 6(1:1), January 2010.
- [BBJ⁺08] P. Bouyer, T. Brihaye, M. Jurdziński, R. Lazić, and M. Rutkowski. Average-price and reachability-price games on hybrid automata with strong resets. In *FORMATS'08*, volume 5215 of *LNCS*, pages 63–77. Springer, 2008.
- [BBL04] P. Bouyer, E. Brinksma, and K. G. Larsen. Staying alive as cheaply as possible. In *HSCC 2004*, volume 2993 of *LNCS*, pages 203–218. Springer, 2004.
- [BBL08] P. Bouyer, E. Brinksma, and K. G. Larsen. Optimal infinite scheduling for multi-priced timed automata. *Formal Methods in System Design*, 32(1):2–23, February 2008.
- [BBM06] P. Bouyer, T. Brihaye, and N. Markey. Improved undecidability results on weighted timed automata. *Information Processing Letters*, 98(5):188–194, 2006.
- [BBR05] T. Brihaye, V. Bruyère, and J.-F. Raskin. On optimal timed strategies. In *FORMATS'05*, volume 3821 of *LNCS*, pages 49–64. Springer, 2005.
- [BCFL04] P. Bouyer, F. Cassez, E. Fleury, and K. G. Larsen. Optimal infinite

- scheduling for multi-priced timed automata. In *FSTTCS*, volume 3328 of *LNCS*, pages 148–160. Springer, 2004.
- [Bel57] R. Bellman. *Dynamic Programming*. Princeton University Press, 1957.
- [Ber01] D. P. Bertsekas. *Dynamic Programming and Optimal Control (Volume 2)*. Athena Scientific, 2 edition, 2001.
- [BFH⁺01] G. Behrmann, A. Fehnker, T. Hune, K. G. Larsen, P. Pettersson, J. Romijn, and F. Vaandrager. Minimum-cost reachability for priced timed automata. In *HSCC'01*, volume 2034 of *LNCS*, pages 147–161. Springer, 2001.
- [BHPR07] T. Brihaye, T. A. Henzinger, V. Prabhu, and J.-F. Raskin. Minimum-time reachability in timed games. In *ICALP'07*, volume 4596 of *LNCS*, pages 825–837. Springer, 2007.
- [BJSV10] J. Berendsen, D. J. Jansen, J. Schmaltz, and F. W. Vaandrager. The axiomatization of override and update. *Applied Logic*, 8(8):141–150, 2010.
- [BLMR06] P. Bouyer, K. G. Larsen, N. Markey, and J. I. Rasmussen. Almost optimal strategies in one-clock priced timed automata. In *Foundations of Software Technology and Theoretical Computer Science, FST&TCS 2006*, volume 4337 of *LNCS*, pages 345–356. Springer, 2006.
- [BM05] T. Brihaye and C. Michaux. On the expressiveness and decidability of o-minimal hybrid systems. *Journal of Complexity*, 21(4):447–478, june 2005.
- [BMRT04] T. Brihaye, C. Michaux, C. Rivière, and C. Troestler. On o-minimal hybrid systems. In *HSCC'04*, volume 2993 of *LNCS*, pages 219–233. Springer, 2004.

- [BSS89] L. Blum, M. Shub, and S. Smale. On a theory of computation and complexity over the real numbers: NP-completeness, recursive functions and universal machines. *American Mathematical Society. Bulletin. New Series*, 21(1):1–46, 1989.
- [BV07] H. Bjorklund and S. Vorobyov. A combinatorial strongly subexponential strategy improvement algorithm for mean payoff games. *Discrete Applied Mathematics*, 155(2):210 – 229, 2007.
- [CHR02] F. Cassez, T. A. Henzinger, and J.-F. Raskin. A comparison of control problems for timed and hybrid systems. In *HSCC’02*, LNCS, pages 134–148. Springer, 2002.
- [Chu62] A. Church. Logic, arithmetic and automata. In *Proceedings of the International Congress of Mathematicians*, pages 23–35, 1962.
- [CY92] C. Courcoubetis and M. Yannakakis. Minimum and maximum delay problems in realtime systems. In *CAV’92*, volume 575 of *LNCS*, pages 399–409. Springer, 1992.
- [dA03] L. de Alfaro. Quantitative verification and control via the mu-calculus. In *CONCUR’03*, volume 2761, pages 103–127. Springer, 2003.
- [dAFH⁺03] L. de Alfaro, F. Faella, T. A. Henzinger, R. Majumdar, and M. Stoelinga. The element of surprise in timed games. In *CONCUR’03*, volume 2761 of *LNCS*, pages 144–158. Springer, 2003.
- [dAHM01] L. de Alfaro, T. A. Henzinger, and R. Majumdar. Symbolic algorithms for infinite-state games. In *CONCUR’01*, volume 2154 of *LNCS*, pages 536–550. Springer, 2001.
- [FV97] J. Filar and K. Vrieze. *Competitive Markov Decision Processes*. Springer, 1997.

- [Gen05] R. Gentilini. Reachability problems on extended o-minimal hybrid automata. In *FORMATS'05*, volume 3829 of *LNCS*, pages 162–176. Springer, 2005.
- [GTW02] E. Grädel, W. Thomas, and T. Wilke, editors. *Automata Logics, and Infinite Games. A Guide to Current Research*, volume 2500 of *LNCS*. Springer, 2002.
- [Hen96] T. A. Henzinger. The theory of hybrid automata. In *LICS 1996*, pages 278–292. IEEE Computer Society Press, 1996.
- [HHM99] T. A. Henzinger, B. Horowitz, and R. Majumdar. Rectangular hybrid games. In *Conference on Concurrency Theory, CONCUR 1999*, volume 1664 of *LNCS*, pages 320–335. Springer, 1999.
- [HK99] T. A. Henzinger and P. Kopke. Discrete-time control for rectangular hybrid automata. *Theoretical Computer Science*, 221:369–392, 1999.
- [HKPP98] T. A. Henzinger, P. W. Kopke, A. Puri, and Varaiya P. What’s decidable about hybrid automata? *Journal of Computer and System Sciences*, 57:94–124, 1998.
- [Hod97] W. Hodges. *A Shorter Model Theory*. Cambridge University Press, 1997.
- [HWT95] P. Ho and H. Wong-Toi. Automated analysis of an audio control protocol. In *CAV'95*, volume 939 of *LNCS*, pages 381–394. Springer, 1995.
- [HWT96] T. A. Henzinger and H. Wong-Toi. Using HyTech to synthesize control parameters for a steam boiler. In *Formal Methods for Industrial Applications*, volume 1165 of *LNCS*, pages 265–282. Springer, 1996.
- [JLR09] M. Jurdziński, R. Lazić, and M. Rutkowski. Average-price-per-reward

- games on hybrid automata with strong resets. In *VMCAI'09*, volume 5403 of *LNCS*, pages 167–181. Springer, 2009.
- [JT07] M. Jurdziński and A. Trivedi. Reachability-time games on timed automata. In *ICALP'07*, volume 4596 of *LNCS*, pages 838–849. Springer, 2007.
- [JT08a] M. Jurdziński and A. Trivedi. Average-time games. In *FSTTCS'08*, volume 08004 of *Dagstuhl Seminar Proceedings*. IBFI, 2008. Available from: <http://drops.dagstuhl.de/opus/volltexte/2008/1765>.
- [JT08b] M. Jurdziński and A. Trivedi. Concavely-priced timed automata. In *FORMATS'08*, volume 5215 of *LNCS*, pages 48–62. Springer, 2008.
- [LMS04] F. Laroussinie, N. Markey, and P. Schnoebelen. Model checking timed automata with one or two clocks. In *CONCUR'04*, volume 3170 of *LNCS*, pages 387–401. Springer, 2004.
- [LPS00] G. Lafferriere, G. J. Pappas, and S. Sastry. O-minimal hybrid systems. *Mathematics of Control, Signals, and Systems*, 13(1):1–21, 2000.
- [LTMM02] S. La Torre, S. Mukhopadhyay, and A. Murano. Optimal-reachability and control for acyclic weighted timed automata. In *IFIP'02*, volume 223 of *TCS*, pages 485–497, 2002.
- [MM97] K. Meer and C. Michaux. A survey on real structural complexity theory. *Bulletin of the Belgian Mathematical Society.*, 4(1):113–148, 1997. Journées Montoises (Mons, 1994).
- [NOSY93] X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. An approach to the description and analysis of hybrid systems. In *Hybrid Systems*, volume 736 of *LNCS*, pages 149–178. Springer, 1993.

- [OMBT03] M. Oishi, I. Mitchell, A. Bayen, and C. Tomlin. Hybrid system verification: Application to user interface design. *to appear in the IEEE Transactions on Control Systems Technology*, 2003.
- [Put94] M. L. Puterman. *Markov Decision Processes. Discrete Stochastic Dynamic Programming*. Wiley, 1994.
- [PV94] A. Puri and P. Varaiya. Decidability of hybrid systems with rectangular differential inclusions. In *CAV'94*, volume 818 of *LNCS*, pages 95–104. Springer, 1994.
- [RLJ10] M. Rutkowski, R. Lazić, and M. Jurdziński. Average-price-per-reward games on hybrid automata with strong resets. *Software Tools for Technology Transfer*, pages 1–17, 2010.
- [Rut11] M. Rutkowski. Two-player reachability-price games on single-clock timed automata. In *QAPL'11*, volume 57 of *EPTCS*, pages 31–46, 2011.
- [RW89] P. J. G. Ramadge and W. M. Wonham. The control of discrete event systems. *IEEE*, 77:81–89, 1989.
- [Tar51] A. Tarski. *A Decision Method for Elementary Algebra and Geometry*. University of California Press, 1951.
- [Tho95] W. Thomas. On the synthesis of strategies in infinite games. In *STACS'95*, volume 900 of *LNCS*, pages 1–13. Springer, 1995.
- [TMBO03] C.J. Tomlin, I. Mitchell, A.M. Bayen, and M. Oishi. Computational techniques for the verification of hybrid systems. *Proceedings of the IEEE*, 91(7):986–1001, 2003.
- [vdD98] L. van den Dries. *Tame Topology and O-Minimal Structures*, volume 248 of *LMSLNS*. Cambridge University Press, 1998.

- [VPVD08] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. Dullerud. Stormed hybrid systems. In *ICALP'08*, volume 5126 of *LNCS*, pages 136–147. Springer, 2008.
- [VPVD09] V. Vladimerou, P. Prabhakar, M. Viswanathan, and G. Dullerud. Stormed hybrid games. In *HSCC'09*, volume 5469 of *LNCS*, pages 480–484. Springer, 2009.
- [Wei00] K. Weihrauch. *Computable Analysis*. Springer, 2000.
- [WTH92a] H. Wong-Toi and G. Hoffman. The control of dense real-time discrete event systems. Technical report, Stanford University, 1992.
- [WTH92b] H. Wong-Toi and G. Hoffman. The input-output control of real-time discrete event systems. In *IEEE Real-Time Systems Symposium*, pages 256–265, 1992.
- [ZP96] U. Zwick and M. Paterson. The complexity of mean payoff games on graphs. *Theoretical Computer Science*, 158(1–2):343 – 359, 1996.